

Automated Dog Ball Launcher: For Post-Covid Dogs

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Ji Sun Alyce Hong

Spring, 2022

Technical Project Team Members

Alexander Byrd

Andrew Childers

Hayden Sarpong

Austin Turner

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Harry C. Powell Jr., Department of Electrical and Computer Engineering

Technical Report

Decapacitors / Doggie Ball Launcher

Alexander Byrd, Andrew Childers, Ji Sun Hong, Hayden Sarpong, Austin Turner

December 17th, 2021

Capstone Design ECE 4440 / ECE4991

Signatures

Alexander Byrd

Andrew Childers

Ji Sun Hong

Hayden Sarpong

Austin Turner

Statement of Work

Alexander Byrd

My primary contribution to the development of the automatic ball launcher was the design of the power PCB. I researched the components necessary for assembling the circuitry and interfacing it with both the microcontroller and the mechanical parts. I found, read, and extracted relevant information from dozens of datasheets to identify the ideal component candidates. I would then select and order the appropriate electrical components for the circuit. I designed circuit schematics in Multisim. With assistance from Andrew, I also created custom footprints for each component that was not in the NI Master Library. Then I uploaded the custom component models and footprints onto a new database on a shared cloud. I designed the PCB layout in Ultiboard and drew the traces connecting each component. I produced the Gerber files used to order the PCBs. With some assistance from Hayden and Andrew, I ordered PCBs from manufacturers and ensured their assembly at the local board assembly.

At the beginning of the project, I researched the underlying physics behind the ball launching mechanisms and calculated the minimum torque and rpm needed to launch a tennis ball by five feet. I assisted Austin and Andrew in interpreting the results of hardware tests. I procured access to the 3D printing machines in the mechanical engineering building, and I cleaned the larger prints with hydrogen peroxide to remove unwanted residue. I procured parts for construction of the chassis.

Andrew Childers

I first created and tested the Bluetooth connection code for the MSP432 using the CC2650 which would allow the microcontroller to communicate with the webserver. I also assisted Alex with the creation of the PCB footprint, which would connect our microcontroller with the motors and servo while powering all the components of the project.

My main contribution to the creation of the automatic dog ball launcher was testing and the construction of the mechanical device. Austin and I tested all the mechanical parts which include the motors, servos, MSP432, CC2650, and our PCB. After all the mechanical parts were tested individually, the mechanical parts were tested together with the issues that arose being fixed by myself and Austin. Then once the mechanical parts were tested together, I took the lead in the creation of the wooden chassis that would house the electronics and support the servo that moved the motors. I measured and cut all the wood used for the chassis and with assistance from Austin constructed the chassis including the parts that were designed by Hayden and 3D printed. Once the chassis was created, I wired the mechanical parts in the final chassis. At the very end I also stained the wood, giving it a nice finish.

Ji Sun Hong

My main contributions to the project's system include the web application and client application programming interface (API) connection. The web application was designed by using React.js and JavaScript along with VSCode and GitHub for version control. I have limited experience using React and JavaScript from my past summer internship. With this limited knowledge, I designed the application by starting off with HTML code and confirming each part was displaying on the application. Then, I implemented each component with React and its hooks. JavaScript was used to write each of the hooks and their executed actions. For the final deliverable, three dropdowns and a submit button were used to send information for distance, direction, and the time set to run the launcher.

The next contribution was the piece of connecting the web application to the Bluetooth API created by Hayden. The API call was done with the use of Axios, an HTTP client for JavaScript applications. With Axios, I made a function which created a POST request to send along the request payload. The request payload information comes from the values selected for each dropdown for distance, direction, and time after the submit button is pressed. The POST request information was then sent to Hayden's Bluetooth API, which would eventually send this same information to the microcontroller to run the ball launcher.

Hayden Sarpong

My main contributions to our automatic dog ball launcher are 3D designing through solid works and the creation of Bluetooth software that connects our website to the hardware of our launcher. The 3D design was conducted through SolidWorks which I have a little bit of experience using through personal experimentation. SolidWorks allowed me to create our main chassis with entry and exiting holes, a motor mount component that allows for the mounting of our launching motors, an exit ramp that launches our racquetball at a 45-degree angle, and a 90-degree entry ramp with a funnel attached to allow for the ball to reach the motor mount in our main chassis. Each of these components were carefully measured to integrate with each other and accommodate the racquetball rolling through the application. The material they are constructed with is plastic as it is highly water resistant.

My other contribution was the Bluetooth connection software. This software was created through the combination of Bleak, a Bluetooth low-energy python package, and an asynchronous function in python. The basic pathway of this program is that once it receives the distance, direction, and time from our website it connects to our microcontroller through a Bluetooth low energy signal. Then our server will send that data to the microcontroller through its Bluetooth write characteristic. From there the microcontroller takes those values and controls our launcher.

Austin Turner

The main contributions that I made to our project were through the microcontroller code. I was responsible for integrating past files into the embedded project that we created and creating

all the new code. The main integration that occurred related to the motor/pulse width modulation and servo code, which we received from Professor Delong and integrated into our project to fit our servo's needs. These previous and new files were integrated through testing to function with our motor driver boards, servo and motors themselves to properly create our ball launching mechanism.

Moreover, I created new code in our main file using Bluetooth code previously written by Andrew to set up a timing system for the launching mechanism, set the servo position, and set the motor speed in a continuous loop based on the data received from the Bluetooth connection. This code functions through receiving a Bluetooth connection, receiving a message and then setting multiple variables. I also integrated timer three to act as a global timer and used a variable that counts seconds to ensure that the timing mechanism was as accurate as possible. This code was all written in C in Code Composer Studio.

The next major contribution to the project that I made came in the form of hardware testing and chassis setup. Andrew and I were the two primary members who tested all the hardware parts with the MSP432. This included the servo, motors, motor driver boards and all the previous parts that were discarded/unused in the final product. This was critical in ensuring that the electronics and hardware functioned properly and was necessary for the completion of our design. Andrew and I also constructed the chassis and I worked with him to assemble the device. This included soldering two prototype boards that I made and constructing the structure that held our device together to ensure that our connections were as stable as possible.

Table of Contents

Contents

Capstone Design ECE 4440 / ECE4991	2
Signatures.....	2
Statement of work:.....	3
Table of Contents.....	6
Table of Figures	7
Abstract.....	9
Background.....	9
Constraints	11
Design Constraints	11
Economic and Cost Constraints	12
Environmental Impact.....	12
Sustainability.....	13
Health and Safety	13
External Standards	13
Tools Employed.....	14
Ethical, Social, and Economic Concerns	15
Intellectual Property Issues.....	15
Detailed Technical Description of Project.....	16
Project Timeline.....	40
Test Plan.....	41
Final Result.....	47
Costs.....	50
Future Work	50
References.....	52
Appendix.....	56
Appendix A – Microcontroller Code	56
Appendix B – Bluetooth Software Code	60
Appendix C – Web Application Code	Error! Bookmark not defined. 58
Appendix D – Cost.....	60

Table of Figures

Figure 1 MSP432P401R with BoostXL-CC26250MA	17
Figure 2 HS-805BB servo.....	18
Figure 3 775 DC Gear Motor.....	18
Figure 4 Pololu G2 High-Power Motor Driver Board.....	19
Figure 5 Power Distribution Circuit	21
Figure 6 High Current Output 6V Linear Regulator Circuit.....	22
Figure 7 5V Linear Regulator Circuit.....	22
Figure 8 Circuit Schematic	23
Figure 9 Final PCB	23
Figure 10 Half of Main Chassis	24
Figure 11 90 Degree Entry Ramp	25
Figure 12 Motor Mount	26
Figure 13 Exit Ramp.....	26
Figure 14 Bluetooth API Flowchart.....	28
Figure 15 Server Connection Established.....	29
Figure 16 Server Bluetooth Initialization	29
Figure 17 Select Device Function and Connection.....	29
Figure 18 Device Hex Write	30
Figure 19 Web Application View	30
Figure 20 Request Payload of Web Application.....	31
Figure 21 Main Firmware Flow Chart.....	33
Figure 22 Intended Circuitry for Solenoid Hatch	35
Figure 23 Considered Closed Loop Motor Control	36
Figure 24 Intended Power Distribution Circuit	37
Figure 25 Intended 3.3V Linear Regulator.....	37
Figure 26 Pin Functionality for Intended Design	38
Figure 27 Intended Pins Selected for Driver Boards	38
Figure 28 Intended Pin Connection for Driver Boards.....	38
Figure 29 Intended Jumpers for Quality-of-Life Additions.....	39

Figure 30 Intended Final Circuit Schematic	39
Figure 31 Intended Final PCB	40
Figure 32 Original Project Timeline	40
Figure 33 Updated Project Timeline Prior to Midterm Design Review	41
Figure 34 Measurement of 5V regulator output	43
Figure 35 Measurement of 6V regulator output	43
Figure 36 Measurement of 12V output to motors.....	44
Figure 37 Measurement of the 6V regulator input.....	44
Figure 38 L78 Regulator Circuit Inspiration	44
Figure 39 Double Checked 6V Regulator Voltage.....	45
Figure 40 5V Jumper Voltage.....	45
Figure 41 3.3V Jumper Voltage.....	45
Figure 42 Servo Signal without Motor Running.....	46
Figure 43 Servo Signal with Motor Running.....	46
Figure 44 Servo Signal with Lowpass Filter.....	47
Figure 45 Servo Signal with Voltage Divider.....	47
Figure 46 Motor Initialization Code	56
Figure 47 Method to Move the Motors Forward	56
Figure 48 Method to Initialize the Pulse Width Modulation Signals	57
Figure 49 Method to Update the Servo Position.....	57
Figure 50 Main Code	59
Figure 51 Server URL Post Path.....	60
Figure 52 Bleak Connection Class.....	61
Figure 53 Connection Manager Function	61
Figure 54 Connection Connect & Select Device Function.....	62
Figure 55 User Console Manager	62
Figure 56 Server Run Function.....	63
Figure 57 Distance, Direction, and Time Hooks	63
Figure 58 Submit Button Hooks	63
Figure 59 POST Request Function	64

Abstract

The automated doggie ball launcher is a system for dog owners or even dog sitters to keep their dog busy and active while they cannot. With the slow shift back into normal office life, the mass dog adoption which happened over quarantine leaves dogs at home alone without their owners. Automating a ball launcher and implementing various features for the distance and direction of the ball allows the dog to teach itself how to play on its own. The device is to be placed outside and connects with an application accessible from any device connected to the internet. Personal devices can connect over Bluetooth with the automated ball launcher allowing the user to enter distance, direction, and time. The idea that sets apart this ball launcher from the rest of the automated ball launchers on the market is the ability for the user to set a certain range of direction along with the rotation of the base. This range of direction with rotation gives a sense of predictability but is aiming to keep the dog engaged instead of being able to predict the same spot a ball will be thrown. The ball will be launched forward with the use of motors and the direction changes due to the servo. The distance of the ball changes depending on the speed the motors launch the ball.

Background

As the season of quarantine slowly starts to phase out and life starts to become more normal, those who once worked remotely will now have to go back into the office, leaving their dogs at home. During the coronavirus pandemic quarantine period, families and dog owners had a chance to interact with their dogs more than they had before. Shelters could not meet the high demand for dog adoptions and sales [1]. The shift back to normality for dog owners leaves dogs alone at home without their families to play with them. To combat this boredom for dogs, an automated ball launcher will be built. Unlike other automated ball launchers on the market, this launcher will be used to teach the dog to play with itself. An application can be accessed by the human user to schedule the ball launcher for a certain time throughout the day. The goal of this launcher is for the dog to play with the launcher and learn to play fetch. The dog will get exercise and a temporary relief from being home alone. The technical project deliverable is a working automated dog ball launcher.

As adults start to head back into the office, dogs will be left alone at home again back to pre-pandemic conditions. Since the 19th century, dogs have been standing steadfast beside their human counterparts [2]. Helping to hunt, acting as guard dogs, and cultivating unbreakable bonds that will last a lifetime. However, owning and caring for a dog is no easy task. With 48,255,413 households having dogs owned as pets in 2018 in the US, entertainment and pet health has been a priority for millions of Americans [3]. Dogs require physical activity, as do humans, and need care and attention. The American Kennel Club recommends that as a general guideline, most dogs need at least two hours of social time with either humans or other dogs every day [4]. While two hours seems like a manageable amount, the busy nature of life sometimes makes it difficult to achieve this daily necessity for your pet, which can lead to

problems at home. Dogs with too much alone time may be destructive, pace or pant uncontrollably, urinate or defecate in the house or excessively bark or howl due to boredom, lack of social stimulation, incomplete training, or more serious conditions like separation anxiety [5]. The coronavirus pandemic made this social time with dogs and humans a lot easier as dog owners had to stay at home, but the amount of interaction will change as dog owners go back to the office.

With this complex situation in mind, the technical project, in cooperation with the four other members, will be designed to provide dogs having to deal with substantial amounts of alone time a form of entertainment to try and prevent conditions like separation anxiety. The project will provide dogs with the ability to play fetch with themselves based on a scheduling system setup through a mobile application by the owner, which will alleviate some of the stress on a pet that spends most of their day alone. The main aim of the project is the dogs will be able to play with itself with the utilization of the automated launching device.

Upon investigation, there are multiple products out on the market that aim to accomplish a similar task as this project. These include but are not limited to the iFetch, PetSafe Automatic Dog Ball Launcher and the iDogMate Ball Launcher. Each one of these mechanisms has defining features and characteristics that separate them from the rest of the market. The iFetch seems to use motors and even has a model that lets the ball roll out randomly in multiple holes around the chassis, which makes for more mental stimulation and unpredictability [6]. On the other hand, the iDogMate includes a motion sensor that detects if someone's pet is standing directly in front of the launcher, which will reduce the risk of injuries as the balls will not launch unless someone's pet is safely clear of the firing zone [7]. However, they all are lacking an advanced scheduling system like this project will have, and none rotate on an axis to fire in a semi-circular manner. Having the axis fire in a semi-circular manner sets this product apart from other competitors in the market and gives it distinct competitive advantages.

The overarching system will be split into three main categories of work: hardware/structural design, embedded software and coding and mobile application development. CAD and the Bluetooth application sections of the system are the most challenging parts as they do not call on any of the previous coursework. The embedded design along with the motor and servo control coding systems will call on the Embedded and Robotic Systems, EARS, and Fundamentals of Electrical Engineering, FUN, series coursework. A part of the system is the closed loop control which allows the ball to launch at different velocities, creating different distances and variations of the launch. The C code to control the servo system will be used to control the device, which calls on former knowledge of C from EARS. Power management for the motors and servos utilized knowledge from Electromagnetic Energy Conversion. Bluetooth code for the microcontroller was used in the EARS course and utilized within the project. As for the web application section of the project, the version control using GitHub and basic understanding of the software development cycle calls on the Advanced Software Development

course. The API code written in Python calls on the first computer science course, CS 1110, in which Python was taught.

Constraints

Design Constraints

The immediate design constraints related to the system were requirements to have a mechanism to rotate the ball, another mechanism to launch the ball, needed a Printed Circuit Board (PCB) and a microcontroller capable of Bluetooth connection to talk with the webserver.

CPU Limitations

The microcontroller chosen for the automatic dog ball launcher is the MS432P401R [8] and the Bluetooth capabilities came from the BOOSTXL-CC2650MA [9]. The MSP432 was chosen because it had four 32-bit timers and two 16-bit timers, which were critical for pulse width modulation signals that needed to be sent and the scheduling mechanism, and the familiarity of the device to many members of the Decapacitators [8]. The BOOSTXL-CC2650MA device was chosen as the Bluetooth extension due to its simple integration with the MSP432. The MSP432 provided plenty of timers which is perfect to properly send all the signals the hardware needed.

Software Limitations

React.js was chosen as the main framework for the front-end development of the website application. Using React.js limited the Bluetooth Low Energy to utilizing an API because the main package of Bluetooth Low Energy made for React applications was for React Native. React Native is used in mobile applications and not web applications, therefore, an API for Bluetooth had to be chosen. Python was decided as the main software to host the Bluetooth code due to the desire to host a server separate to the website host. Through a bit of research, the Python Bluetooth package Bleak was found to not only connect to Bluetooth low energy systems like Arduinos but also had the ability to read and write information through the services provided by the devices. Python also allowed for the establishment of a local server through an Ipv4 address and port. The Fastapi and Uvicorn python packages allowed the software to all be contained on the server and run through the Bluetooth connection code once a network request was received.

Hardware Limitations

One of the major constraints for the hardware was part accessibility as there were many delays in procuring parts due to supply chain shortages or delays. As a result, damaging or determining that the hardware was insufficient meant that a replacement would not arrive for another week. The parts available produced additional limitations. One of the most notable limitations is that the wall adapter used to power the motors is rated for 5A while the motors can draw up to 12A each. The PCB design is also constrained by the adhering to the necessary dimensions to be placed on top of the microcontroller. Another constraint is that the servo selected had to be able to support the weight and torque of the motors and the launching apparatus. Therefore, the chassis had to be designed in a manner to conserve horizontal space to reduce the torque.

Economic and Cost Constraints

The Decapacitors were only allocated \$500 for this project, which led to the obvious constraint of not being able to buy extremely expensive parts for the system. However, the main limitation occurred in the fact that the group could not buy multiple of the same part, especially more expensive parts like the motor driver boards. While it would have been ideal to have backups of everything that was being worked on in case anything drastic were to occur, the budget would not allow for multiple parts to be purchased with this safety factor in mind. Overall, the budget was not a serious limiting factor, and the group was able to work around this limitation.

Environmental Impact

The main environmental concern for this project is the use of plastic, as it is one of the main materials used for the 3D printed pieces. Use of plastic is an environmental concern due to its inability to biodegrade, which leads to a surplus of the material being in landfills. This project will use as little plastic as possible to limit waste contribution. The 3D printed pieces used polylactic acid (PLA) plastic, which is biodegradable [10]. The limited plastic for the overall system and the biodegradable 3D printed pieces keeps harmful environmental costs low. Other than the use of batteries and the use of plastic, this project's environmental handprint is low since its intended use is only for an hour a day [11]. Another environmental concern for this project is having wood as the main housing material for the electronics. Currently, logging makes up ten to fifteen percent of deforestation across the world. With the intention of mass producing the automatic dog ball launcher, this would lead to a greater contribution to the demand for logging. Contributing to logging was unavoidable for this project as the encasing needed to house electronics was not 3D printable. As a result, wood was selected as an alternative housing material [12]. Lastly, printed circuit boards are not easily recycled and often contribute to a lot of electronic waste in landfills across the nation. The PCB was unavoidable as it is the main source of power for the system's motors and servo.

Sustainability

With the current system design, the lifetime of the launcher is dependent on how long the microcontroller can last. The system's main concern with sustainability is the microcontroller used has been discontinued [8]. The other parts of the system, including the wooden base, servo, plastic chassis casing and plastic motors can all be easily replaced. Specifically, the chassis casing, launching tube, and plastic motors were all 3D printed, and can be reproduced with the appropriate files and 3D printing devices. Besides the main sustainability concern with the microcontroller, the next piece would need more replacements depending on the conditions the launcher is stored under. If kept in damp and wet conditions, the wooden base is capable of rotting within the day. As for the 3D printed pieces, time would be the only hinderance in the way of sustainability.

Health and Safety

Physical activity for a dog is important and the project is aimed at maintaining and assisting the health of a dog. As for animal safety of the project, a main concern is the material being used for the automated ball launcher. Currently, there are no safety standards set by the government for regulating dog toys, but the goal of the project is to keep harmful materials from being accessible to the dog [13]. To achieve this goal, the hardware is encased to prevent the dog from being able to have direct access to potentially harmful substances.

Safety for human interaction with the device, more specifically the PLA plastic 3D printed pieces, is deemed safe. With a classification as a toy, there is a safety standard addressing how phthalates are prohibited [14]. PLA plastic used in the plastic chassis, motors and launching tube are phthalates free, therefore, safe and within the safety standard for toy interaction [10].

External Standards

1. National Electrical Manufacturers Association (NEMA) and International Protection Marking (IP) IP44's water standard standards that deals with different mechanical and electrical casings that align to the project to protect the product from water damage was followed to the level of creating a chassis that is water resistant [16].
2. NEMA's Motor and Generator Operational standard 7.7.3.2 will be followed, and this states direct current machines shall be supplied with the armature voltage and field current corresponding to the speed at which vibration is being measured [17].
3. NEMA's Motor and Generator Operational standard 7.7.3.3 states unless otherwise specified for machines having more than one fixed speed the limits of this part shall not be exceeded at any operational speed [17].

4. NFPA (National Fire Protection Association) 70 Article 430 which focuses on motor safety to prevent fires and other dangers. This article goes into different details about motor circuit design standards and its components like how a 30 – 60-minute rated motor has a short duty cycle of 150 and an intermittent duty cycle of 90 [18].
5. Institute of Electrical and Electronics Engineers (IEEE)'s 802.15.1-2002's standard for telecommunications and information exchange between systems was taken into consideration as our server transmitted information to our microcontroller locally [20].
6. Embedded C Coding Standard by the Barr Group was taken into consideration when constructing the microcontroller code. This was used to model the new code written and all old files that the group had written and integrated into the project [20].

Tools Employed

Firmware

All the firmware that was written in this project was done using Code Composer Studio [16] in the C programming language for our MSP432 microcontroller [21]. There are also multiple header files written for the TI-RSLK MAX [22] education kit by Dr. John Valvano. These header files were integrated into the written code to make the launcher function properly.

There was also additional code that interacted with the BOOSTXL-CC2650MA [9] which was the Bluetooth module used to connect with the web server that was developed. This code was created with Code Composer Studio like the rest and was written in C.

Software

All the Bluetooth software was created through Python 3.9, an interpreted, interactive, object-oriented programming language [23]. This language helped to create the necessary algorithms to receive information from a website, establish a connection with a MSP432, and send that information as interpretable information to a MSP432. The following python packages installed through pip also heavily supported the creation of this software:

- Bleak, a GATT client software that can connect to BLE devices acting as GATT servers [24]
- Aioconsole, an asynchronous console and interfaces that provides support for async input and output functions [25]
- FastAPI, a web framework for building APIs with Python 3.6+ based on standard Python type hints [26]
- Uvicorn, an ASGI server implementation, using uvloop and httptools [27]

Visual Studio Code, a lightweight but powerful source code editor, comes with built-in support for JavaScript, Python, HTML, and Node.js, which were languages used to write all the software [28]. Github, a web-based Git repository hosting service offering all the distributed revision control and source code management functionality, was used to host different versions of software [29]. Using Github allowed multiple team members to work on the code in parallel. The basis for the web application front-end was React.js [30]. For the React.js application, the following packages were installed:

- React-Bootstrap, a Bootstrap Javascript framework rebuilt for React.js [31]
- Axios, a promise-based HTTP client for the browser and Node.js, utilizing the http module and the XMLHttpRequests [32]

Ethical, Social, and Economic Concerns

One of the main ethical concerns is the idea of animal neglect and the mobilization of human functions. The reasoning for a dog to learn how to play with the device on its own is to aid the dog owner in keeping the dog active. To maintain their health, dogs need to maintain a certain threshold of activity. There are some dog owners that may not physically have the capability to play fetch with their dogs for how much exercise and activity the dog needs, and this project serves to assist. Based on how much this device would cost, another economic concern would be product availability for all levels of income. This device would be a luxury due to the expensive parts that were used, and it would be difficult to make it available for all individuals at a cheap and affordable price.

Another Ethical concern for this project is the harm that can come to either the dog owner or actual dog. This machine uses high power motors to launch the ball in different directions and due to that ability, it is possible for a significant injury to happen if someone were to be hit by the ball. Especially depending on where the animal may be hit at, it could lead to broken bones, concussions, and contusions. This device allows the dog owner to set the speed and direction at which it is traveling to hopefully prevent this sort of situation but there is still a possibility of it happening.

Intellectual Property Issues

The automatic ball launcher referenced in [33] consists of a housing mechanism with a launching hammer on the inside. The system has a platform with pivot connections on either side of the housing, enabling the angle of the ball launch to change based on the turning of these connections. The launching mechanism is completely different from the Decapitators' launching mechanism as [33] references a hammer with a sensor. The hammer and sensor ensure the machine only fires the launching mechanism when the ball is placed properly in the shaft. iFetch also has a pivoting structure which allows the angle of exit to be changed when the ball is

launched. The pivoting structure simulates variation with different distances and angles for the dog's entertainment. All items considered, the launcher within the iFetch device is vastly different from the Decapacitators' system. This is not likely to impose any issues with intellectual property as the differences of the mechanics and the lack of the ability to rotate 180 degrees to fire the ball at any angle, which is one main feature of the Decapacitators' project.

The next device is described as a pet operated ball thrower and is referenced in [34]. This device consists of "a base, a fulcrum post, attached at its lower end to the base, a throw arm pivotally attached to the upper end of the fulcrum post, a ball receiving receptacle attached to the throw arm adjacent its outer end, a treadle pivotally attached at its lower end to the base, and an attachment member pivotally attached at its, lower end to the treadle and pivotally attached at its upper end to the throw arm." The operation entails the pet jumping on the base which causes the arm to spin around to throw the ball forward for the pet to fetch and return device to put the ball in the receptacle for the process to repeat. Again, the Pet Operated Ball Thrower is extremely far from the Decapacitators' ball launcher and is in no way similar in the operation, features and the launching mechanism. This patent would provide no problem with intellectual property claims to the Decapacitators' device and did not include the main rotating feature that was intentionally designed to alleviate intellectual property conflicts.

The final device in [35] is a handheld pet throwing mechanism which is supposed to assist owners who may not be able to throw the ball themselves or get tired easily when doing so in long sessions. The ball thrower device has two ends. One end responsible for throwing the ball and another end reserving a ball for the owner to switch into the throwing end when ready. The arm acts as an extension of the individual and is swung in a throwing motion, either underhand or overhand, to extend the distance the ball is thrown. As the ball launcher acts as an extension, the device makes the process of throwing a ball easier in the physical aspect. Another main feature of the device is how it folds together for easier storage, to make the space it takes up more compact. This mechanism will not cause any intellectual property conflicts with the Decapacitators' launcher because it is supposed to be launched by the owner and provides support for this rather than being an "automated" dog ball launcher.

Based on the patents surveyed which were available in the field, there are some with similar aspects of the automatic ball launcher created by the Decapacitators, but there are none containing the main feature of rotating 180 degrees to fire a ball in various angles, or a similar housing mechanism.

Detailed Technical Description of Project

The intended project was an automatic dog ball launcher which allows the pet owner to set the specific specifications of distance fired, direction fired, and time ran for through a web application. This web application would be hosted on a separate server and once the parameters for distance, direction, and time are set, the values are sent to an API endpoint. The API endpoint

takes the values set by the application user and triggers the start of the Bluetooth low energy discovery software. This software finds all the Bluetooth low energy devices in the area and connects to the microcontroller for the launcher. Once the connection to the microcontroller is established, the values received would be converted to hex format and combined. This converted hex message is sent as bytes to the microcontroller. These bytes are interpreted in their hexadecimal format bit-by-bit to change the values of the motor speed, servo direction, and overall runtime. After values have been set in the microcontroller, signals are sent to the necessary hardware components of the servo and motor to rotate the automatic dog ball launcher and run the motors. A racquetball is placed on the entry ramp and rolls down until it reaches the main motors. The racquetball is compressed by the motor on either side and is launched out through the exit ramp.

Hardware

A MSP432P401R in combination with a BOOSTXL-CC2650MA as seen in Figure 1 is the system that firmware is installed on.

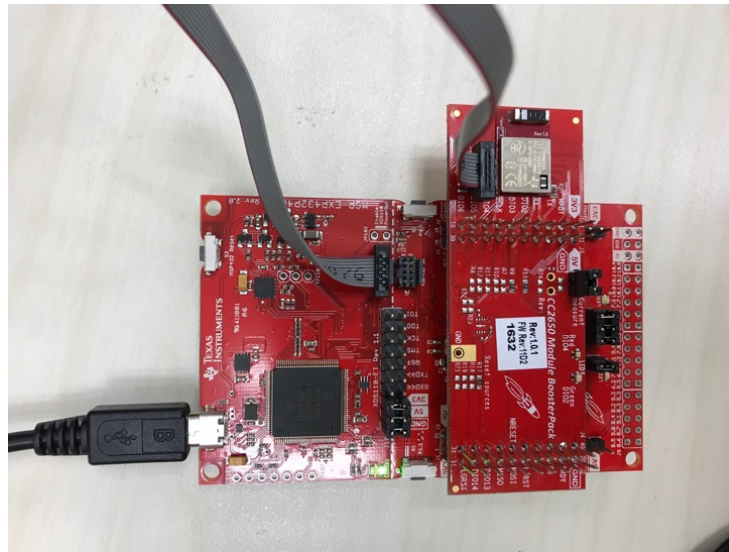


Figure 1 MSP432P401R with BoostXL-CC26250MA

To integrate the CC26250 Bluetooth module onto the microcontroller, female headers were added onto the bottom of the module and placed on the male headers of the MSP432. Doing this allowed the Bluetooth module to be powered by the microcontroller. The combination of the Bluetooth module and the microcontroller acted similarly to the brain of the launcher which is responsible for controlling the components described in the remainder of the hardware section.

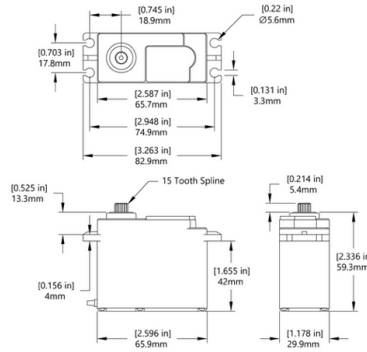


Figure 2 HS-805BB servo

The HS-805BB servo was utilized to rotate the chassis of the ball launcher. The HS-805BB servo was selected because it is capable of rotating 199 degrees. Its capability for rotation fulfilled the goal of developing a ball launcher capable of rotating 180 degrees. Another benefit of this servomotor was it was readily available equipment in Charlottesville and did not require a significant wait to obtain. Furthermore, the HS-805BB servo had a stall torque of 343.01 ounce-inches while at its rated six volts. The specified stall torque is proven experimentally sufficient for rotating the weight of the chassis on top of it using experiments described in the test plans section. No stall torque was provided for the servo motor, but it has a no-load current of 830mA at its rated 6V operating voltage. Therefore, for safety reasons precautions were taken to ensure that the circuitry could handle the servos if they delivered a large current. [36]

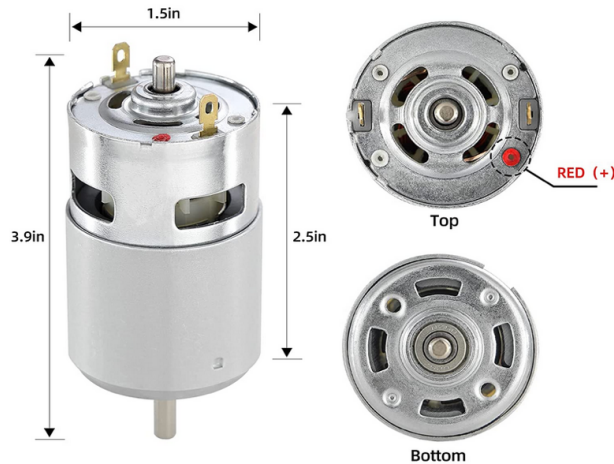


Figure 3 775 DC Gear Motor

Two 775 DC Gear Motors were selected due to their versatility and wide range of reachable rotations per minute (RPM). With a 12 Volt supply, the 775 DC gear motors could

reach up to 10,000 RPM. The 775 DC Gear Motor is rated at 12 to 24 volts with current ratings between 4 and 12 amps. Additionally, they are advertised to have high torque while only weighing 0.634 ounces each, so the servo can support their weight if placed directly above it with minimal offset. The high torque and RPM of the motors also implied that they could easily squeeze a racket ball between them to launch it a substantial distance. An additional feature of the 775 DC gear motors is how they are quieter than most motors when provided with a high frequency PWM. [37]

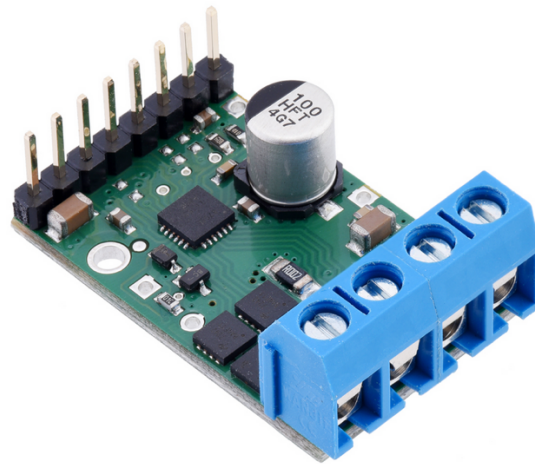


Figure 4 Pololu G2 High-Power Motor Driver Board

Two Pololu G2 High-Power Motor Driver Board were selected to control the speed of each motor. The G2 driver boards can handle an input voltage between 6 and 30 Volts while supplying up to 17Amps continuously to the motors. The high amp and voltage rating meant that the motors would never be able to draw more current than the driver board could supply. The G2 driver board also offered numerous safety features such as fault detection and current sensing pins accessible by the microcontroller. Additionally, it was able to accept PWM signals of up to 100kHz, which meant it could achieve the 20kHz necessary to make the motors run with minimal audible noise. It also provided a 3.3V output pin so that it could supply additional low-current circuitry in future additions to the design. [38]

A Facmogu brand 12V 5A wall adapter was selected as the power supply for the system. While 5A is not able to run the motors at the higher range of their RPM, it is still capable of providing them with enough current to launch the ball several feet [40]. The wall adapter was selected primarily due to its economic viability and accessibility. This 12V 5A wall adapter costs \$13, but one capable of supplying the 20Amps necessary to access the higher range of RPM values the 775 motors are capable of are around \$50 dollars and are in short supply [39]. The current rating of this wall adapter serves as one of the primary constraints in the distance that the motors can launch the ball [40]. Luckily, the wall adapter has built in overcurrent protection, so it simply shuts off when too much current is demanded. When it shuts off the current, a blue light

on it began flashing, which was helpful for debugging purposes [40]. The wall adapter is connected to the circuitry via a power jack coupling rated for the same amperage as the wall adapter.[41].

Circuit Design

The components utilized to construct the circuitry for this project are in Table 1. Their placement is illustrated in Figure 5.

Table 1: Component parts in final product PCB.

Ref Des	Description	Link
U1	L7805ABD2T-TR linear 5V regulator	https://www.digikey.com/en/products/detail/stmicroelectronics/L7805ABD2T-TR/585694
U2	L7806CD2T-TR linear 6V regulator	https://www.digikey.com/en/products/detail/stmicroelectronics/L7806CD2T-TR/1884018
R1	2.7 Ohm Current Sense Resistor	https://www.digikey.com/en/products/detail/susumu/R1632R-2R70-F/714376
R2	3 home resistors	https://www.digikey.com/en/products/detail/susumu/R1632R-3R00-F/714377
C1 , C3	CAPACITOR, 0.33 μ F	https://www.digikey.com/en/products/detail/kemet/C1206C334K3RAC7800/416048
C2 , C4	CAPACITOR, 0.1 μ F	https://www.digikey.com/en/products/detail/kemet/C1206C104J1RAC7800/2215064
Q1 , Q2	ZXTP25020DFHTA, PNP BJT	https://www.digikey.com/en/products/detail/diodes-incorporated/ZXTP25020DFHTA/1557752

J3	TERMINAL_BLOCKS, 282834-2	https://www.digikey.ca/en/products/detail/te-connectivity-amp-connectors/282834-2/1150135
J1, J2	HEADERS_TEST, 1-534206-0	https://www.digikey.com/en/products/detail/samtec-inc/SSQ-110-23-L-D/8093543
J5	HEADERS_TEST, HDR1X6	https://www.digikey.com/en/products/detail/sullins-connector-solutions/PPTC061LFBN-RC/810145
J4	TERMINAL_BLOCKS, 282834-5	https://www.digikey.com/en/products/detail/te-connectivity-amp-connectors/282834-5/1153266?s=N4IgTCBcDa4BxjgZgCwFoCsIC6BfIA

The circuit is designed to accommodate this hardware is intended to distribute the power from the wall adapter to the motors, the servo, and the microcontroller. Power distribution begins at the wall adapter. The motor driver boards receive power directly from the wall adapter, while everything else utilizes voltage regulators due to operating at a lower voltage.

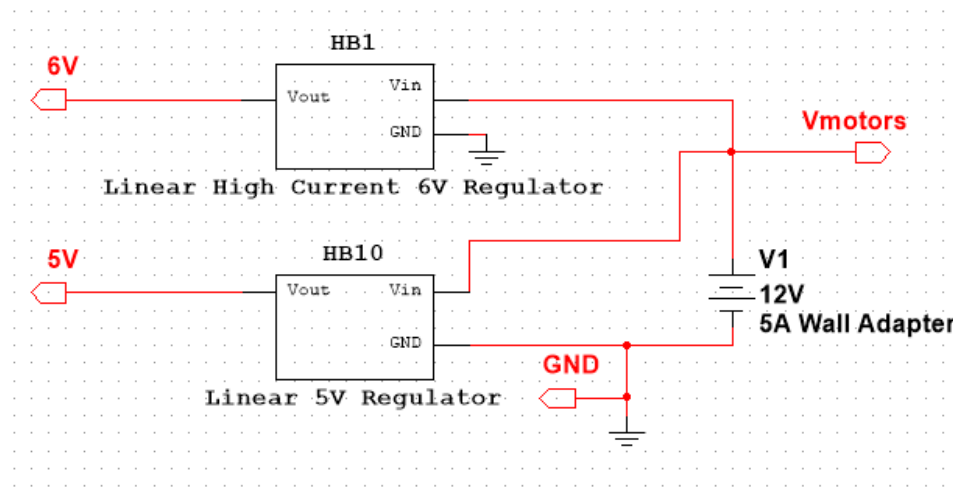


Figure 5 Power Distribution Circuit

A high current 6V linear regulator delivers power to the servo. The resistors and the transistors at the input of the 6V regulator are recommended circuit components for if the output of the regulator requires more than 1.5Amps [42]. Additionally, this circuit is meant to provide short circuit protection. This was implemented for the servomotor since its minimum current draw is rated to be 830mA and the current it would have to draw was unknown as the chassis

was not fully designed at the time. As a precaution, the possibility of a large current draw was accounted for with this circuit. Unfortunately, the circuit ended up causing a short instead of protecting from one due to Q1 having too low of a threshold voltage [42] [43]. Q1 was removed from the board, while the remaining R1, R2, and Q2 remained since they had a negligible impact on the regulator. Additionally, it was anticipated that a newer PCB model would have arrived before the demo.

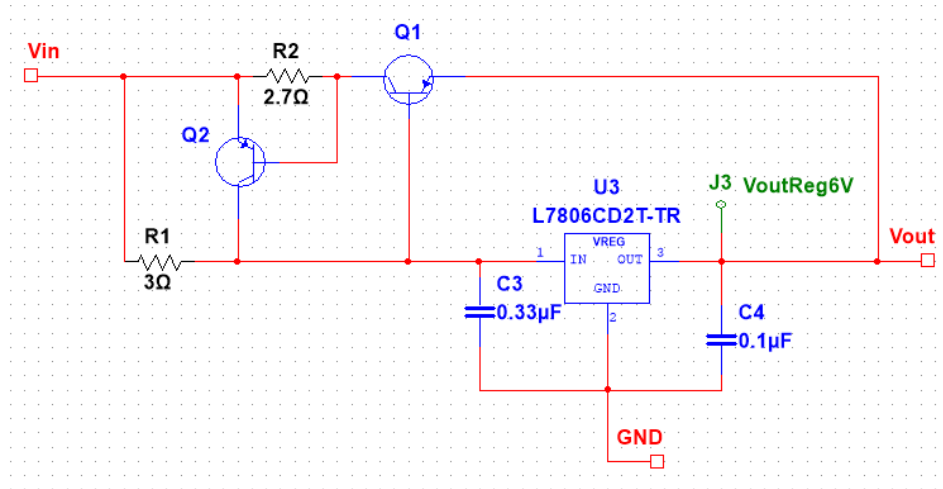


Figure 6 High Current Output 6V Linear Regulator Circuit

A 5V Linear regulator delivers 5V to the 5V pin of the microcontroller. Due to difficulties finding the datasheet referenced later in the problems and redesigns section, this 5V regulator is entirely unused. [44]

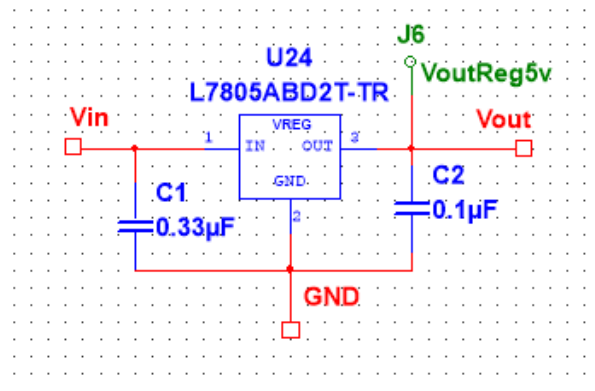


Figure 7 5V Linear Regulator Circuit

The circuit is intended to connect the pins on the microcontroller to the appropriate pins on the hardware it is controlling. However, an issue occurred somewhere when transferring files

between team members before printing the PCB. J2 on figure 8 becomes flipped along the y axis when implemented into the PCB in figure 9. This meant that the pin expected to control the PWM of the first motor was grounded, which caused a short every time that PWM1 was held high. This was resolved by temporarily using jumpers, as a new PCB was designed and expected to replace it.

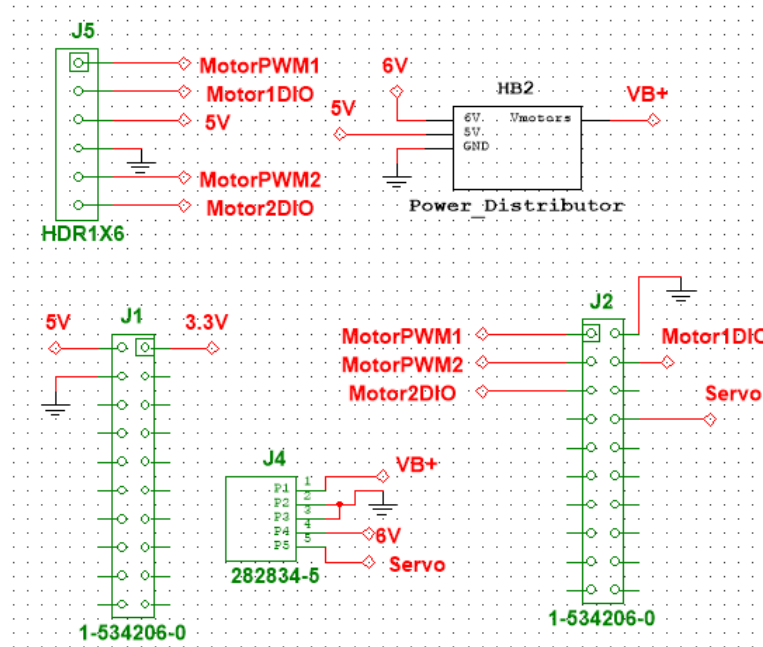


Figure 8 Circuit Schematic

Original circuit PCB design:

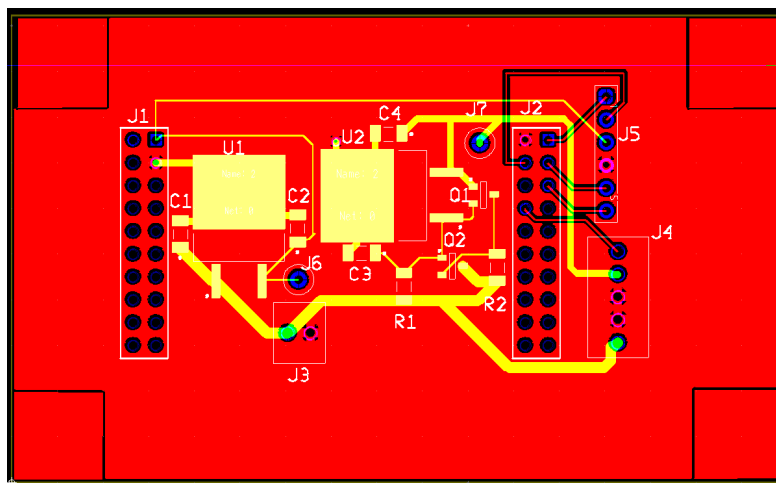


Figure 9 Final PCB

The printed circuit board's job is to provide adequate power to all the components. The power originates from a 12V 5A wall adapter [40] that plugs into a power jack socket [41]. The ground and power pins of the power jack socket are input into the terminal block J3. After power is inputted into the PCB through J3 it is sent to the motor driver boards [38] through terminal block J4. The power is also sent to the 6v regulator U2 [42] which lowers the voltage from 12V to 6V to power the servo [36], which is sent out through the terminal block J4 as well. The bypass capacitor C3 shorts high frequency noise into U2 [45]. The capacitor C4 is responsible for shorting any high frequency noise or feedback from the servo [46]. The four corners of the PCB are removed from the ground plane so that holes may be drilled into them to mount the PCB onto the chassis. The holes needed to be disconnected from the ground power plane in order to ensure that the screws do not conduct current. However, the design presently uses command strips to keep the PCB on the chassis so that it could be easily removed for debugging. U1, C1, C2, Q1, Q2, R1, R2, and J6 are discussed further not used in the final design for reasons discussed further in the problems and design modifications.

While the power comes from the printed circuit board, the signals to control the components are sent from the MSP320401R [17]. The MSP4320401R is responsible for sending a PWM signal to the motors and a special PWM signal to the servo. The PWM signal to the motors comes out of pins 2.6 and 2.7 and is sent to the motor driver boards to the PWM input on the driver board. The direction pin and the ground pin were also grounded to make sure the microcontroller and the driver boards shared the same ground while also ensuring the motors spun the correct way. The other pins on the driver board were unused. With the PWM signal and the power, the driver board sends a 12V PWM signal to the motors. The servo signal was sent out of pin 5.7 passing through a voltage divider using a 6.8k Ohm resistor and the impedance of the servo. After the noise of the signal was reduced, the signal was sent to the servo. To encase the electronics a chassis was designed.

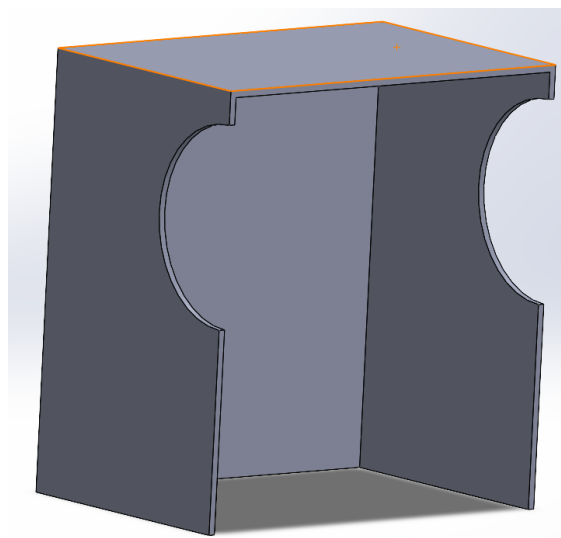


Figure 10 Half of Main Chassis

Half of the chassis depicted, by figure 10, was designed through SolidWorks. Half of the chassis was designed so the components inside could be tested and configured as necessary, before sealing all the motor components inside. The size of the chassis was configured based off the motor mount and motors that would all be contained inside of the whole chassis. The size design also took into consideration the entry and exit ramps' connection to motor mount.

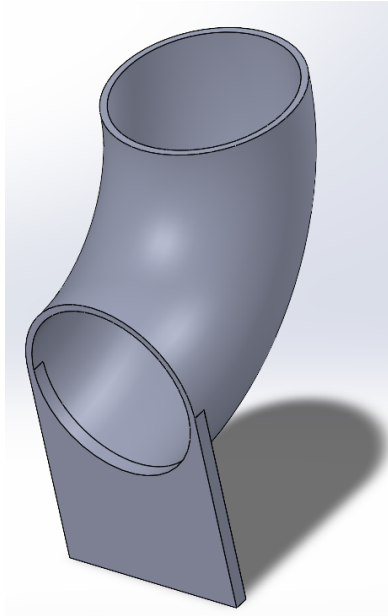


Figure 11 90 Degree Entry Ramp

Figure 11 depicts the entry ramp designed to enter the chassis and connect to the main motor mount part. The entry design to be 90 degrees maximizes the speed the racquetball would enter the motor mount. Having the entry at 90 degrees ensures the motor wheels would catch a compress the ball out. A base plate was added to the bottom of the exit point of the entry ramp to allow for sturdy connection to the motor mount part. The base plate design lets the racquetball roll to the ramp enough so it would be unlikely to miss the entrance of the motor mount.

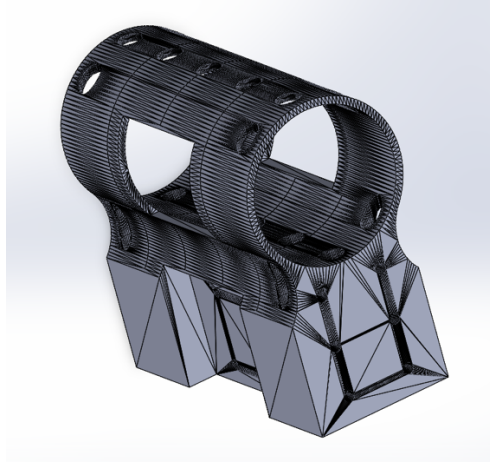


Figure 12 Motor Mount

As mentioned before Figure 12, the motor mount, connects to the entry ramp and harbors the main launcher motors. This CAD design came from an open MakerBot file sharer, Thing Verse [44]. The holes in the main spherical tube allow the motor wheels to enter the tube and compress the racquetball. The base component of the motor mount feature screw holes to connect the entry and exit ramp to the mount so there is smooth passage between the entry point, launch point, and exit point. The base component of the motor mount as contains a divot to attach the motor with its axle facing up.

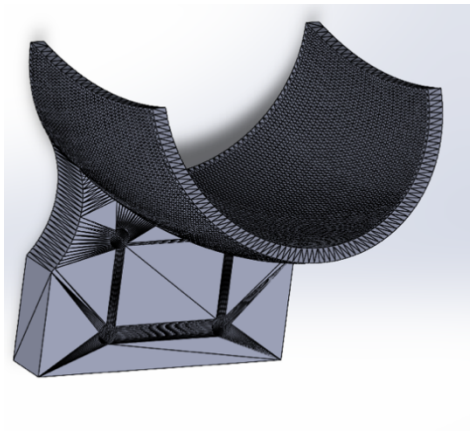


Figure 13 Exit Ramp

Figure 13 above shows the exit ramp that is connected to the exit port of the motor mount. This design was also taken from an open-source CAD design file sharer [47]. This design is not angled at 45 degrees which would give us the best high to launch the racquetball at but provided us with everything else the launcher would need to launch the ball at an angle. This design was also a pair with the motor mount and provided assurance that it would connect to the

motor mount. The rest of the chassis was designed out of wood to enclose the electrical devices and to support the launching mechanism.

Software

Bluetooth Low Energy Software

To connect to the MSP432P401R, a Bluetooth low energy connection had to be established. Bluetooth low energy was used instead of a regular Bluetooth connection because due to its ability to remain in sleep mode unless a connection is initiated [48]. The Bluetooth connection lets there be a short and quick connection with a low data rate. Using Bluetooth low energy also allowed the microcontroller to run using its own power source, as it would not need to constantly be on to receive a connection. The data being sent to microcontroller is not large and does not require extended periods of connection. This means there is no need to run the connection for prolonged periods of time.

Once the type of Bluetooth connection was decided, a Bluetooth low energy python package, Bleak, was used to establish the electronic connection to the microcontroller. In addition to establishing the Bluetooth connection, a server also needed to be created to receive the launching parameters from the web application. Fast API, a python package used to create API frameworks and establish servers, was decided to receive the network request from the web application due to the familiarity of team members and comfortability of creating an API and server to host it [26]. These two packages in conjunction created the connecting point between the website and the microcontroller.

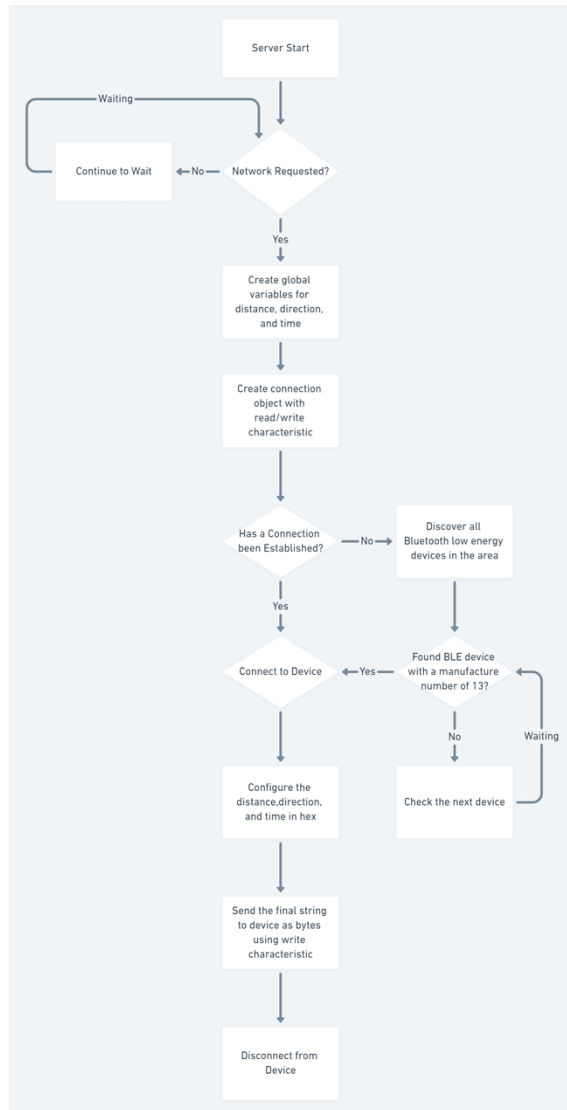


Figure 14 Bluetooth API Flowchart

To initiate the connection between the website and the microcontroller, a server needed to be established. This server was started through a local IPv4 address of 172.25.137.205 and a port of 8000. This local development server was created through Fast API's Uvicorn ASGI server. In addition to the server base URL and addition server path was created that would receive a data model containing a distance, direction, and time. This server path's URL is 172.25.137.205/items. After the server is initiated, it waits to receive a network request on the 172.25.137.205/items URL path. Once the website sends the request, a connection is established between them, and the API function is started.

```

INFO: Started server process [21600]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://172.25.137.205:8000 (Press CTRL+C to quit)
INFO: 172.25.164.186:57562 - "POST /items HTTP/1.1" 307 Temporary Redirect
INFO: 172.25.164.186:57562 - "POST /items/ HTTP/1.1" 200 OK

```

Figure 15 Server Connection Established

The API function then takes the distance, direction, and time received from the body of the network request and creates associated global variables. These global variables are later used as the values written to the microcontroller as hex values. After creating these values, a connection object is created with the read and write characteristic of the microcontroller. These characteristics derive from the different services that our microcontroller can provide and are like receiving permission where the API is only allowed to write or read these services with the correct characteristics provided [49]. The connection then uses these attributes to be able to write and read to these established connections.

After the connection object has been created, the connection object's manager function runs. This function checks if the current connection has already been established with a client. If a connection has already been established the connect function is run to set the connection's connection attribute to true as depicted on the server in Figure 17. If the client has not been established the select device function is run where all the Bluetooth low energy devices in the area are scanned and if there is a device that matches the manufacturer number of our microcontroller it establishes a connection with it.

```

Starting connection manager.
Bluetooth LE hardware warming up...
C:\Users\Hayde\Box Sync\4th Year\Capstone\sam-app\fastapi\app\fastapible.py:139: DeprecationWarning: The loop argument is deprecated since Python 3.8, and scheduled for removal in Python 3.10.
  await asyncio.sleep(2.0, loop=loop) # Wait for BLE to initialize.
C:\Users\Hayde\Box Sync\4th Year\Capstone\sam-app\fastapi\app\fastapible.py:211: DeprecationWarning: The loop argument is deprecated since Python 3.8, and scheduled for removal in Python 3.10.
  await asyncio.sleep(2.0, loop=loop)

```

Figure 16 Server Bluetooth Initialization

```

[]
dict_keys([76])
6:
[]
dict_keys([6])
7:
['0000fddf-0000-1000-8000-00805f9b34fb']
dict_keys([])
8:
[]
dict_keys([76])
9:
[]
dict_keys([13])
C:\Users\Hayde\Box Sync\4th Year\Capstone\sam-app\fastapi\app\fastapible.py:112: DeprecationWarning: The loop argument is deprecated since Python 3.8, and scheduled for removal in Python 3.10.
  await asyncio.sleep(15.0, loop=loop)
C:\Users\Hayde\Box Sync\4th Year\Capstone\sam-app\fastapi\app\fastapible.py:119: FutureWarning: is_connected has been changed to a property. Calling it as an async method will be removed in a future version
  self.connected = await self.client.is_connected()
Connected to

```

Figure 17 Select Device Function and Connection

After the connection to the microcontroller is established, then the connection object's user console manager function is run. This function will check if the connection object has a client and has been connected to that client. If the connection to the client has not been established this function will wait until a connection has been created. If the connection has been created, then an input string is created with the distance, direction, and time values sent through the API call. This string is then written through the connection to the microcontroller as hex byte values and is disconnected from the microcontroller once written to the microcontroller. This connection is disconnected by raising a Bleak client error that forces the whole bleak client to disconnect from all devices.

```
BB0A
Sent: BB0A
Exception in callback BleakClientWinRT.connect.<locals>.handle_connection_status_changed(0) at C:\Users\Hayde\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\bleak\backends\winrt\client.py:207
handle: <Handle BleakClientWinRT.connect.<locals>.handle_connection_status_changed(0) at C:\Users\Hayde\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\bleak\backends\winrt\client.py:207>
Traceback (most recent call last):
  File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.2544.0_x64_qbz5n2kfra8p0\lib\asyncio\events.py", line 88, in _run
    self._context.run(self._callback, *self._args)
  File "C:\Users\Hayde\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\bleak\backends\winrt\client.py", line 216, in handle_connection_status_changed
    self.disconnect_callback(self)
TypeError: on_disconnect() missing 1 required positional argument: 'future'
```

Figure 18 Device Hex Write

Web Application

With the Bluetooth API set up, the next part of the system is the web application. The front-end of the web application used a JavaScript library called React.js and can render on the server using Node.js [30].

The first set up of the website was a pure HTML form with a submit button. There were three input fields which took in a string value for distance, direction, and time. After making sure the user interface was displaying on the local host, the next step was utilizing React components and hooks [30]. To satisfy and consider the changes within the field, a constant called “value” was made with three fields of direction, distance, and time. The useState hook was used to set the values for direction, distance, and time. Having simple input fields which took in any type of string was not restrictive enough for the type of options for the launcher, so the idea of a dropdown for each field was suggested with a choice of given options. Once the dropdowns were implemented, the web application currently looks as it does in Figure 19.

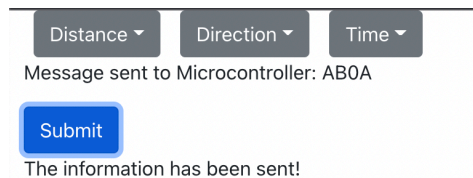
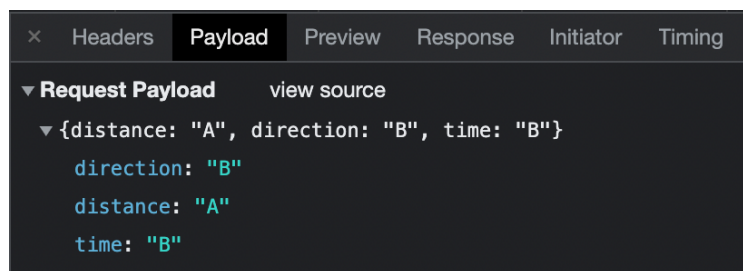


Figure 19 Web Application View

Instead of the text input fields, the format of the website shifted to an overarching form with dropdown buttons within the form. To fit this new format, the hooks were changed. The value constant with three parts for distance, direction, and time were separated into three separate hooks with states and useStates for each constant. This allowed each dropdown to function separately without having to access the same value constantly and only set one part of the value. A dropdown button for distance, direction, and time was made. Each of the dropdown options consists of three options to select from. The distance dropdown button has the options of a short, medium, or long distance. Direction allows the user to select left, right or forward. Time has the three options of ten minutes, twenty minutes, and thirty minutes. Currently, the time options only run for the respective number of seconds instead of minutes. The three options all have a key value of either “A”, “B”, or “C”. Key values were set to these values because this is the format the Bluetooth API needs to eventually send to the microcontroller. Figure 20 shows the request payload of the values for direction, distance, and time, once the submit button has been selected.



```
Request Payload view source
{distance: "A", direction: "B", time: "B"}
  direction: "B"
  distance: "A"
  time: "B"
```

Figure 20 Request Payload of Web Application

To make an HTTP request using Node.js, the axios library was utilized instead of the fetch that is built in. Axios is a helpful library since it automatically converts the data to a JSON format, so it does not have to manually be done [32]. Within the code, axios was used to create a createPost function which creates a post request from the client side of the web application. The baseUrl specified within the function to create a post request was initially set to the Amazon Web Services API link, but it is now Hayden’s computer’s public IP address [50]. This means that the computer the web application is hosted on will connect to Hayden’s computer to access the Bluetooth API on the server side.

The initial design for most of the project duration was to have the Bluetooth API hosted on Amazon Web Services, and the web application would connect the API endpoint hosted on the cloud [50]. However, there were Python package dependencies involved in building the serverless application which caused errors. The next attempt was hosting the web application on Heroku while moving the Bluetooth API to a local server [51]. By hosting the web application on Heroku, there were resulting errors on the Bluetooth API side due to unresolved cloud errors connecting to Bluetooth from Heroku’s cloud.

Firmware

The firmware was first integrated with previous motor and pulse width modulation code from Embedded Computing and Robotics II to fit the motors and motor driver boards that were purchased. The motor code initializes needed pulse width modulation pins and sets the speed of the motors. Originally, the motors utilized an enable and a direction pin, but these are unnecessary since the new system developed pulls certain pins to ground as only one direction is needed for the motors in our launcher. Next, the pulse width modulation code initializes Timer A0 on the MSP432 to utilize capture compare registers three and four for duty cycle value comparison [21]. It also sets pins 2.6 and 2.7 to be primary function module output pins to handle the pulse width signals. Next, the Bluetooth code was created and reimplemented by Andrew to fit the needs of the system using the BOOSTXL-CC2650MA booster pack [9].

The Bluetooth code that was implemented communicates with the BOOSTXL-CC2650MA over a specific set of pins. The CC2650 waits to receive a packet asking for the information and capabilities of the device, which is a hardcoded message since this information does not change. After that, a connection packet is sent, and a handshake is exchanged between the CC2650 and the webserver. Then the webserver sends the message packet containing the firmware data; the motor speed, servo position, and run time. After which, the webserver sends a disconnection packet and the CC2650 sends a disconnection packet back which is also hardcoded message.

The next major implementation was the servo code provided by Professor Todd Delong. The servo that was utilized uses a pulse width modulation signal on pin 5.7 on the microcontroller Timer A2 with capture compare register two. The function worked by taking in an input degree ranging from 0 to 180 and using a linear equation to scale this value into a proper duty cycle. The pulse width modulation signal was then changed on pin 5.7 to make the servo move to that respective degree value.

Finally, the overarching control algorithm was created in the main file that controlled the whole mechanism. This file utilizes all the previous code and multiple TI-RSLK education curriculum files to create a system that constantly searched for Bluetooth connection and messages to set the hardware appropriately [22]. The control begins in an infinite while loop that continuously runs so that the controller is constantly looking for Bluetooth connections. From this, the controller checks to see if a message is received. Once a message is received, the data is deciphered using bit masking to determine what data was sent by the webserver. The first section of the receive buffer is dedicated to both the motor and the servo and the second section handles the timing values. The following table illustrates the corresponding values based on the received value:

Table 2: Received Values and Outputs

Motor Received	Motor Output (Duty Cycle Percentage)	Servo Received	Servo Output (degrees)	Time Received	Time Output (seconds)
0x0A	40%	0x0A	5	0x0A	5
0x0B	50%	0x0B	102	0x0B	10
0x0C	60%	0x0C	180	0x0C	15

These values illustrate the different possibilities that the control algorithm can produce through various checks. Once these checks are completed, the algorithm enters a block of code that sets the servo position and motor speed. Next, a global timer is used that counts seconds to check if the time of duration has been surpassed in a while loop. When the time is up, the code turns the motors off, sets the servo position back to 102 degrees and proceeds to loop back to the top of the algorithm. The overall flow of the structure is illustrated in the figure below.

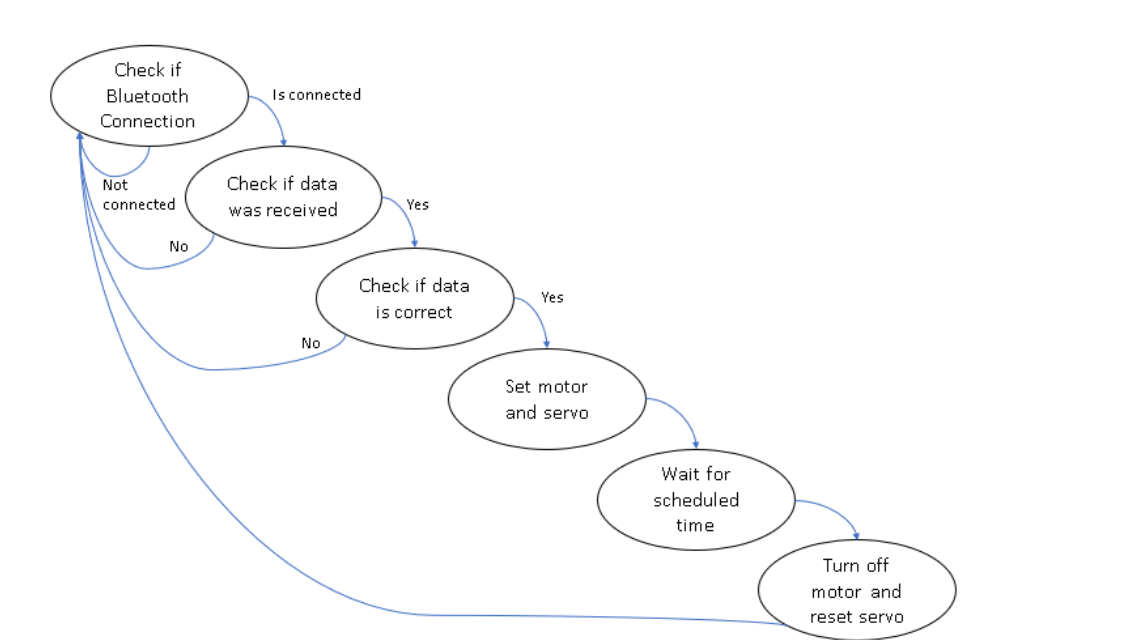


Figure 21 Main Firmware Flow Chart

With this algorithm complete, the firmware and embedded code was ready for implementation into the full system and testing with the hardware. Additionally, the code for all these methods is referenced in Appendix A below.

Problems and Design Modifications

The first fundamental problem occurred with the LAUNCHXL-CC26X2R1 [52]. This device required multiple new cores being installed to professionally install and run the code, which involved downloading multiple items to Code Composer Studio and reinstalling it altogether. After this was done, the device theoretically should have been ready to be flashed and operated with. However, this did not work, and this led to the use of the Cloud version of Code Composer Studio which allowed code to be flashed to the device, but was very inefficient for collaborative, group work [53]. Next, Austin realized that there was not enough clear documentation that would merit the time commitment of learning the new microprocessor, versus using the MSP432P401R and the Bluetooth booster pack which was familiar to him. As such, the microcontroller was swapped out and the coding process continued.

A major unforeseen problem that occurred in Bluetooth Software API was the inability to upload the code onto an AWS Lambda Function due to a dependency problem. A necessary dependency linked to Bleak functionality was unable to be resolved on to the AWS server due to a Pip Build error. As this was unforeseen, it led to an ample amount of time looking through stack-overflow and GitHub forums trying to understand and resolve this problem. To resolve this problem, Docker, an open platform for developing, shipping, and running applications, was installed to create a container that would harbor all the files rather than AWS, but this led to similar dependencies issues. Once AWS was recognized as a being the problem, there were attempts made to upload the API onto Heroku and Okteto which are two other API endpoint creation services that all lead to the same error. In the end, a local server was created that used the IPv4 address mentioned in the Bluetooth Low Energy section to create a server the web application could connect to.

One major setback encountered was that the first set of hardware ordered was insufficient. The motors could not reach a high enough RPM or sustain the necessary torque to launch a tennis ball [54]. The motors were not strong enough to push the ball, so they had to be replaced. Unfortunately, most of the other components had been chosen to work with this motor, which draws significantly less current than the motors that replaced them [37]. However, the old components were still tested with the new motors to conserve the projects budget unless replacements were undeniably necessary.

The first testable hardware that arrived was the 1057 Pololu Servomotor [55]. Unfortunately, the original servo motor was delivered broken. The arm of the servomotor would not rotate consistently and would become jammed even without a load. The servomotor was dismantled to study its gears. It was observed that the gears were improper dimensions and

occasionally their teeth would misalign. Additionally, surplus grease on the exterior of the gears were clumped in volumes large enough to cause occasional jams between the gears. Attempting to remove excess grease and reassemble it proved fruitless as too many of the gears simply had the wrong dimensions and were not symmetrical. A spare HS-805BB servomotor was eventually acquired from the Undergraduate NI Lounge to replace the malfunctioning servomotor [36]

Due to the many unexpected changes and the large delays needed to receive the parts to account for the changes, a few aspects of the design had to be removed in favor of maintaining the project's economic and temporary budgets. The initial design was going to implement a solenoid hatch to prevent the ball from reaching the motors. The solenoid hatch would be opened when it was time to launch the ball, allowing it to roll into the motors and be launched at a controllable time. This would have been beneficial as it would reduce the likelihood of a ball getting jammed and allow for a controllable delay between when the ball is input and when it is launched.

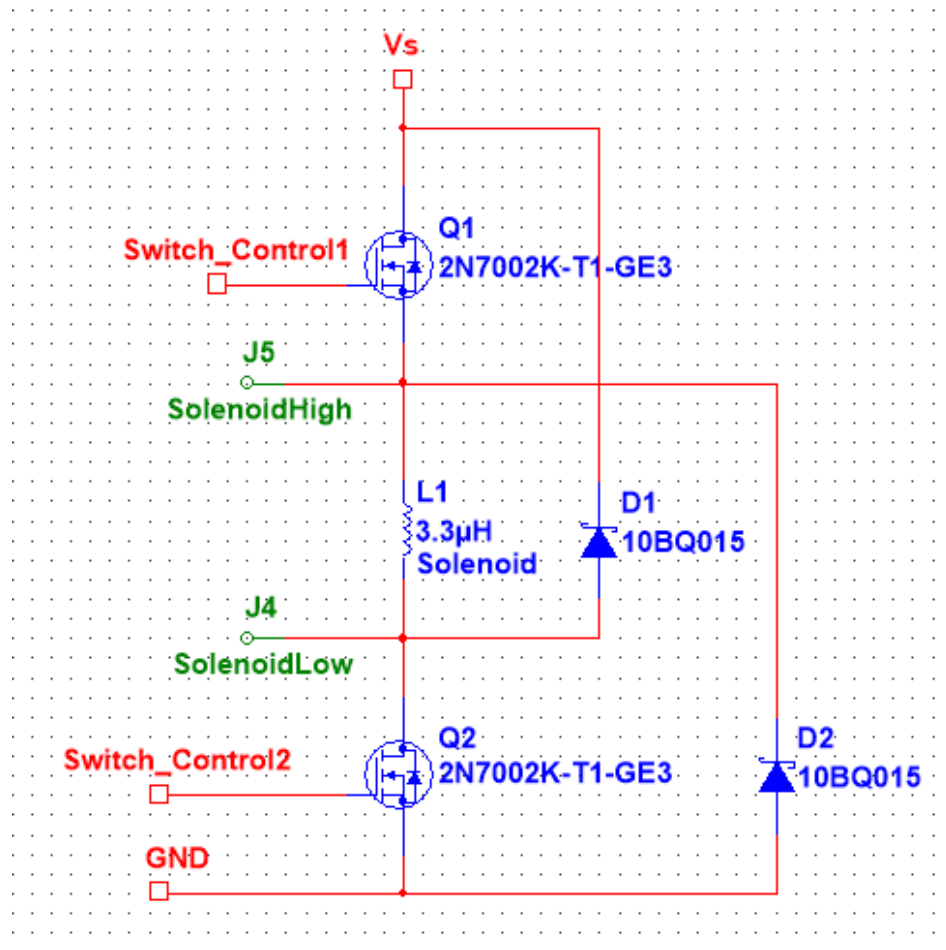


Figure 22 Intended Circuitry for Solenoid Hatch

However, implementing the solenoid hatch had many mechanical implications that became clearer as the project progressed. The solenoid hatch would have to be placed in the tube that the ball is placed into. Unfortunately, that part of the chassis is the farthest part from the center of rotation of the servo. This means that in addition to increasing the servo's torque load, wiring the solenoid hatch to the PCB would risk damaging the PCB. Wires would rotate the entire 180 degrees along the radius of the launching chassis, while the PCB is stationary, which would exert tension on the PCB and the launching chassis. An extremely slack wire would have solved it, but loose wires are not aesthetically pleasing or safe for exposure to pets. Additionally, it would require an extra degree of precision in the 3D printing process, which was already a limiting factor to the design.

Closed loop motor control for the motors was initially intended for the ball launcher. However, using the encoders was considered unnecessary for the purpose of launching a ball. While the encoders would have increased the precision of the motors, the slight increase in consistency could not justify the additional complexity or margin for hardware error [56]. Using the encoders became impractical once the dual driver board was discovered to be inadequate [57]. The new driver board already had feedback capabilities to implement safety features such as current control and fault detection [38].

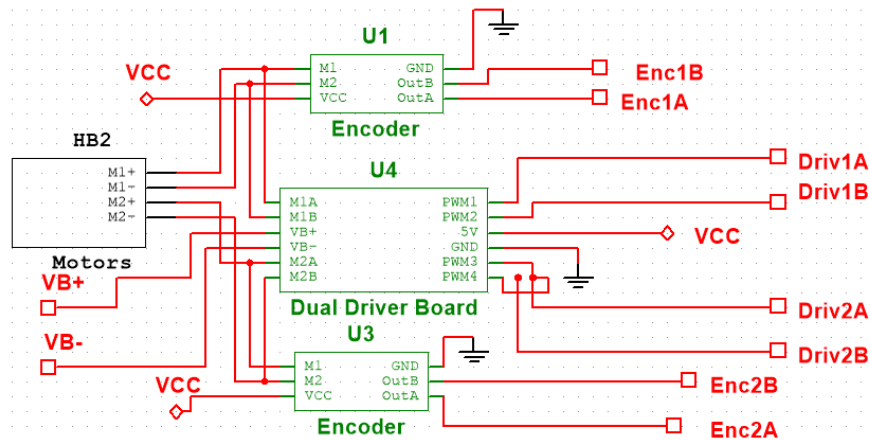


Figure 23 Considered Closed Loop Motor Control

The design of the PCB had been significantly improved, but unfortunately due to supply shortages and shipping delays, it was unable to arrive in time. The power distributor would have utilized a 3.3V regulator instead of a 5V regulator so that it could power the MSP432P401R launchpad with the external power supply. This is because it was not until late into the project that the datasheet for the MSP432P401R was found using archives.org, because TI removed its datasheet from its page and the page of all distributors. Initially, the design was influenced by forums discussing the external powering of the launchpad, but they are inconsistent with the datasheet [58].

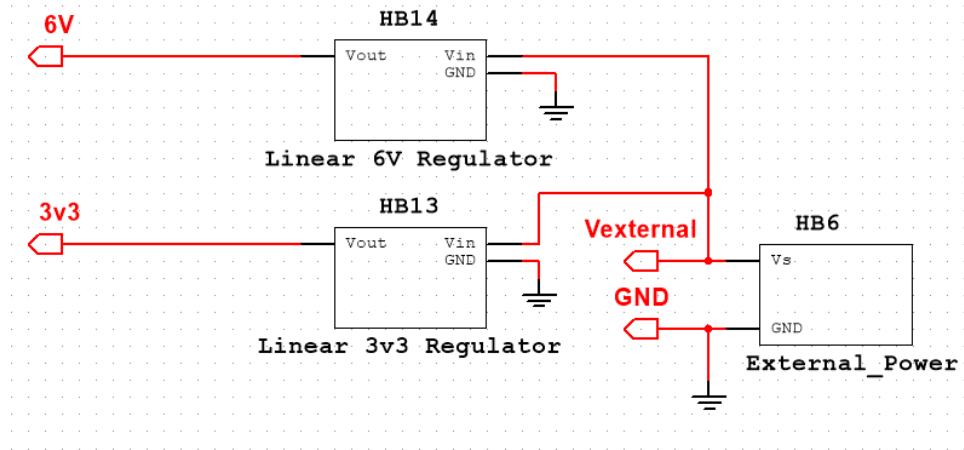


Figure 24 Intended Power Distribution Circuit

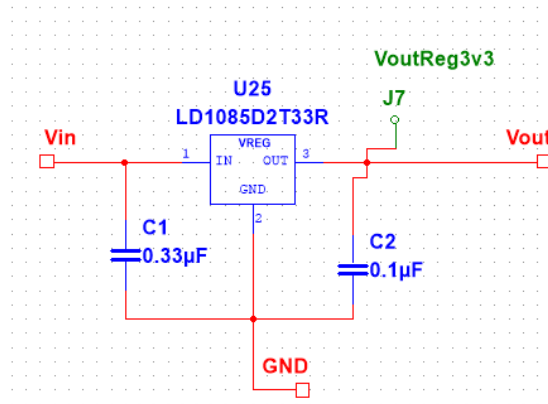


Figure 25 Intended 3.3V Linear Regulator

Additionally, the intended final circuit design included all the connections necessary to the driver boards such that wiring the pins would not be necessary. The pins available to control the driver board were determined using TI’s booster pack compatibility checker [59]. Figures 26 and 27 illustrate the pins that were determined to be available, their functionality, and which were selected to be used with the driver boards.

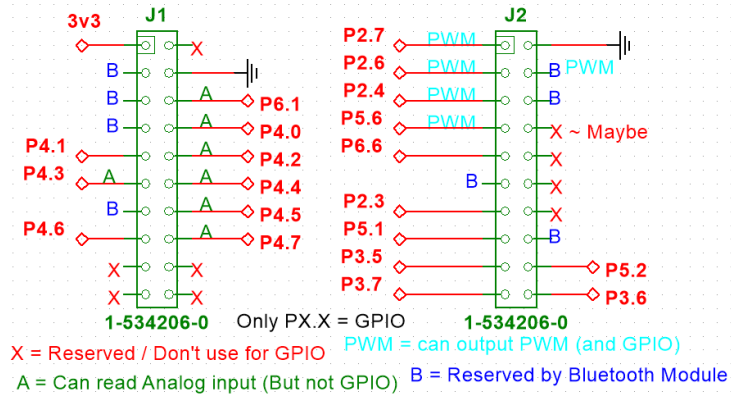


Figure 26 Pin Functionality for Intended Design

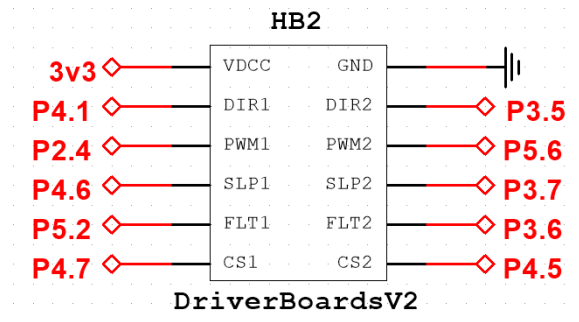


Figure 27 Intended Pins Selected for Driver Boards

The pins are arranged to match the 8pin array that the driver board would attach to the board with via female headers.

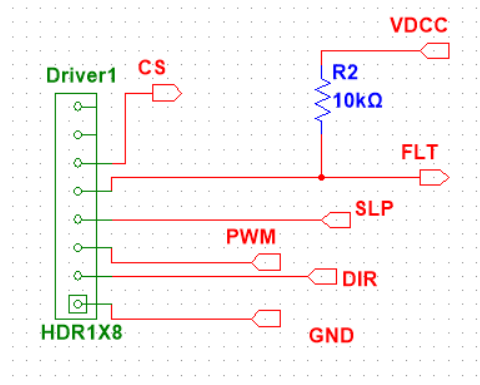


Figure 28 Intended Pin Connection for Driver Boards

Additionally, an external set of jumpers was added to disconnect the microcontroller from the power supply without needing to remove power to the other circuitry also. Another jumper allowed for disconnecting the servo from the digital pins since at the time that the board was ordered the servo had not been tested. An extra pair of terminal blocks was added to connect to the second driver board. The second driver board allows for more flexibility where the driver boards could be placed on the PCB without making a mess of wires.

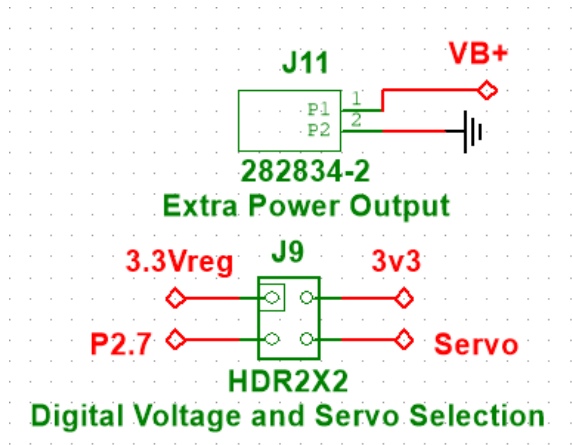


Figure 29 Intended Jumpers for Quality-of-Life Additions

The final circuit schematic for the PCB design that did not arrive in time is shown below in Figure 30. The PCB design itself is shown in Figure 31.

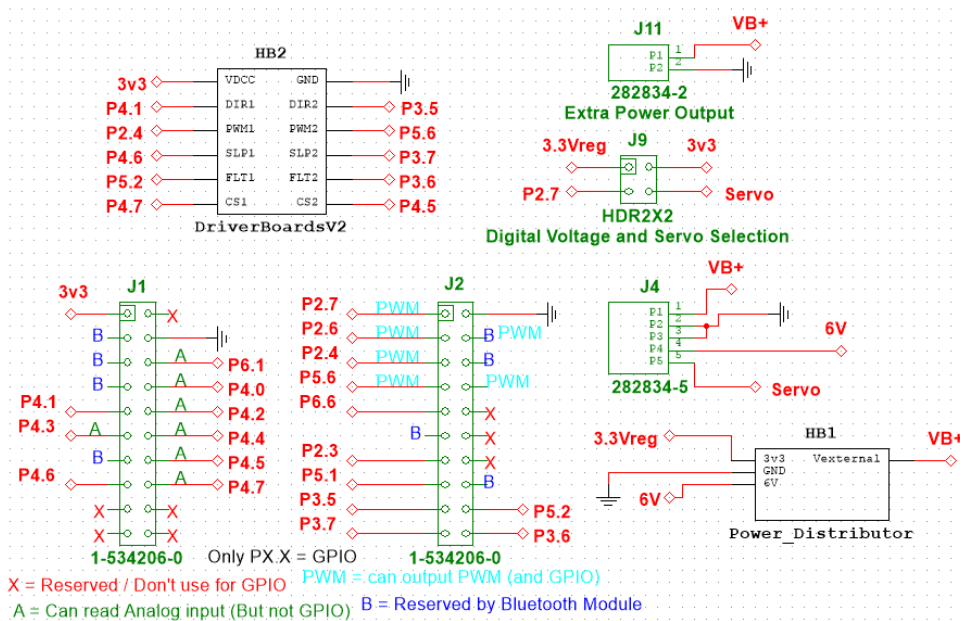


Figure 30 Intended Final Circuit Schematic

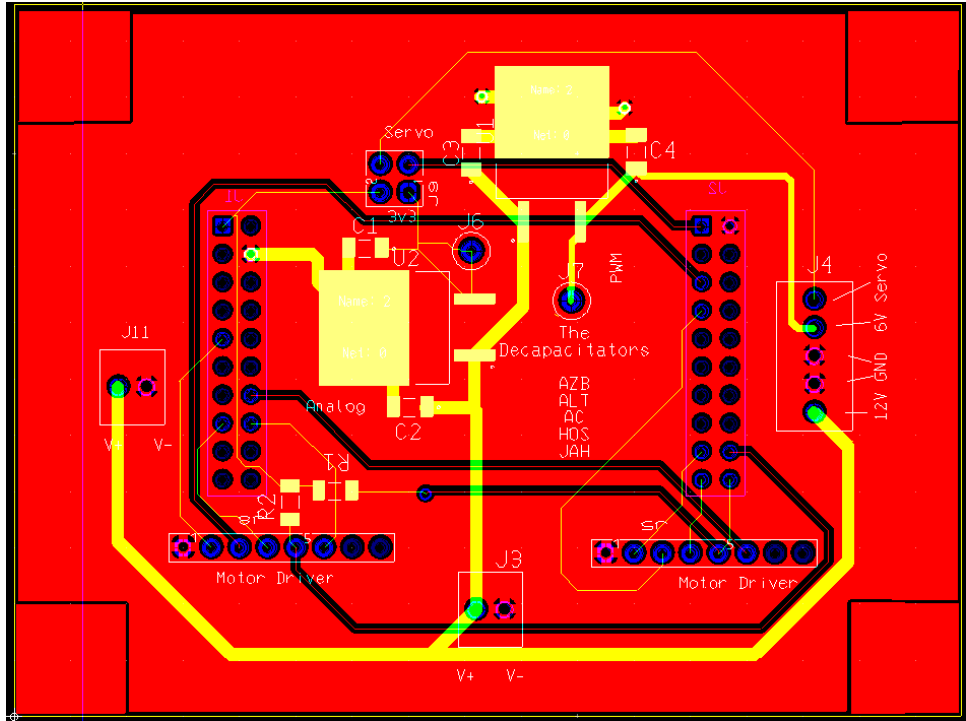


Figure 31 Intended Final PCB

If the intended final PCB had arrived on time, then less wiring would have been needed in the final design and the launcher would be entirely powered with an external supply, and not need an additional USB supply.

Project Timeline

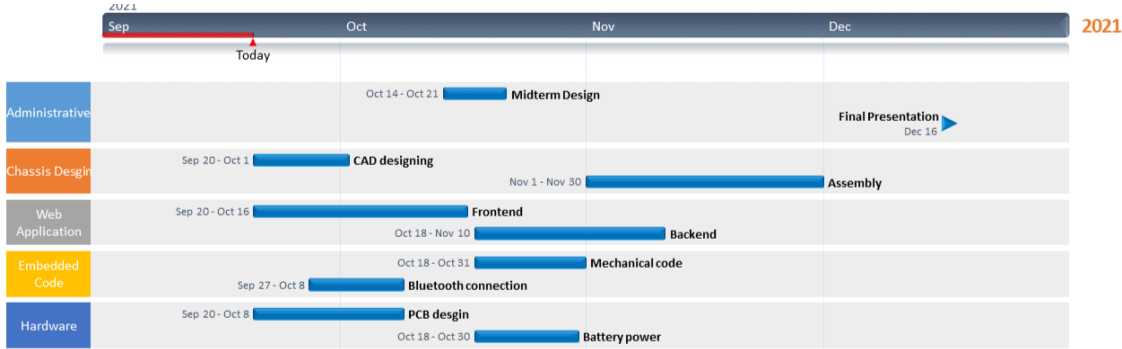


Figure 32 Original Project Timeline

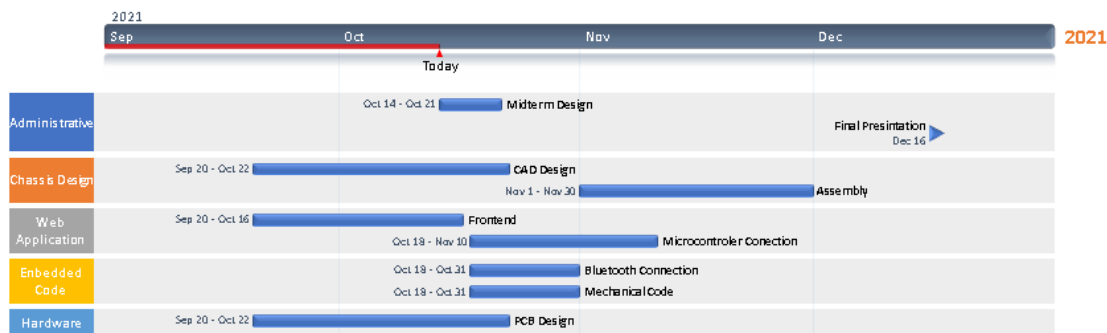


Figure 33 Updated Project Timeline Prior to Midterm Design Review

Office Timeline was used to create the timeline of the project and the beginning of the semester and during the middle of the semester after adding adjustments. The project was divided into major parts such as Chassis Design, Web Application, Embedded Code, and Hardware; with key dates marked under the administrative tab.

Alexander Byrd’s primary role was printed circuit board design, and his secondary role was finding mechanical parts. Andrew Childers’ primary role was embedded coding, and his secondary role will be to the hardware and PCB design and chassis assembly. Alyce Hong’s primary role was Web application development focusing on the frontend with a secondary role assisting the back end of the web application. Austin Turner’s Primary role was embedded code of the project with a secondary role of chassis assembly. Hayden Sarpong’s primary role was Web application development focusing on the backend with the secondary role of CAD design.

Test Plan

The project was divided into submodules for the test plan based on the software, firmware, or hardware being tested. for the test plan. The submodules are Bluetooth compatibility with the microcontroller, Bluetooth accessibility for the server, the servo, the motors, the motor driver boards, circuit combability debugging, launching the ball, and finally testing how the system performed as a whole.

The Bluetooth Software connection was tested actively through a virtual Bluetooth low energy device creation app called LightBlue [60]. This application allowed for the creation of a virtual BLE device with multiple services and characteristics to use. Once the Bluetooth code

was fleshed out, a virtual device was created service that allowed for reading and writing. Then the ability to connect to the BLE device was tested and confirmed through the virtual device. Once the connection was established the virtual BLE device's write characteristic was used to test the write functionality and if the write was sent that value was then read back through the read functionality. This virtual device gave a clear picture of what the actual device was going to be.

The Bluetooth API connection was also tested through the Postman application. Postman is a desktop application that allows for API calls to be made to a HTTP endpoint [61]. After the local server was created, Postman was used to call the local server with a Json body of distance, direction, and time. This call would then be used to test if the server were found and if the body values were received. Once these two results were received then the server was known to work completely.

The servo was tested by observing the degrees of rotation it was able to reach under no load conditions. This verified that it had the mobility advertised. Additionally, it would allow for adjustments of the clock frequency or duty cycle to ensure that the angle produced is consistent with expectations. The PWM of a servo is dependent on the value of the pulse width rather than the percentage of the duty cycle, so the clock period must be greater than the max pulse width. The HS-805BB servomotor had a max pulse width of 2420 microseconds and a minimum pulse width of 556 microseconds, so it was run with a period of 6666 microseconds so that the entire pulse range was available. The servomotor was tested with duty cycles ranging from 9% to 36% to view the nearly 200-degree freedom of rotation.

The torque capabilities of the servo were tested by placing a heavy plank of wood and the chassis on the servo arm. This tested the amount of torque that the servo could support. This test qualitatively verified that the servomotors could support the weight and shape of the launch mechanism [36].

The 2212 Pololu Micro Metal Gearmotors and its included wheels were next to arrive [54]. The functionality of the motors was tested with the square wave of a function generator from the virtual bench. It was verified that the gearmotors would adjust their speed appropriately with the duty cycle of a 6V square wave supplied from a function generator input on the Virtual Bench [62]. The gearmotors were then attached to a temporary launch apparatus that separated them by a little under the diameter of a tennis ball. The motors immediately stalled when a ball was placed between them. They could not provide the necessary torque to squeeze and propel the tennis ball through them.

The new motors were tested in the same manner, but the virtual bench was unable to supply enough current to power one of them [63]. The Virtual bench was limited to a 500mA output while the motors could draw up to 12A [62]. The motors would jolt for a single instant and stop moving in an alarming fashion. Improvised by utilizing 12-volt lead acid battery. The

old 6V power supply was insufficient, and a miscommunication caused a 200mA rating instead of 2A rating to be ordered.

The Maker MDD3A dual driver board was tested using the with the old motor and the virtual bench and things worked fine except for the shortcomings of the old motors. Pulse width modulation was confirmed to control the motor's power supply at the same duty cycle that was input into it [57]. However, when testing the new motors with the Maker MDD3A dual driver board, they could not support the motor as their speed was increased. Even though the wall adapter had overcurrent protection, the was a peak of 8 Amps allowed before cutting off the current. Unfortunately, an instance of 8 Amps was enough to permanently short something on the Maker MDD3A, so it became unusable. The new Pololu G2 High-Power motor driver boards that replaced the Maker MDD3A driver board though were able to run the motors at all the currents that the wall adapter allowed. It was also confirmed with a virtual bench that it would output the expected PWM to the motors.

The voltage regulators on the PCB also had to be verified to determine whether it delivered the desired voltage. The test point J6 on the original PCB design allowed for testing the output of the 5V regulator, L7805ABD2T-TR.

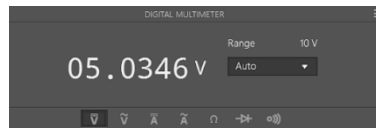


Figure 34 Measurement of 5V regulator output

The test point on J7 was allowed for testing the output of the 6V regulator, L7806CD2T-TR. The 6-volt regulator was malfunctioning. Measurement of the 12 V supply for the motors was performed on the lower pin of J4 to determine whether the 12V supply delivered the specified voltage. Additionally, this would determine whether the 12V and 6V traces were shorted.

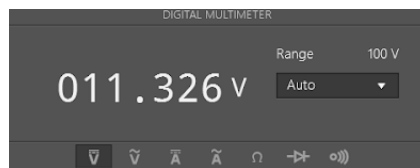


Figure 35 Measurement of 6V regulator output

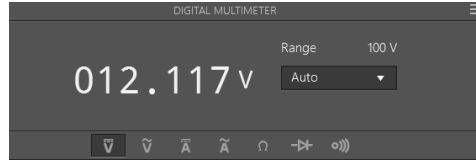


Figure 36 Measurement of 12V output to motors

The measurements in Figure 36 indicated that there was not a direct short between the 6V and 12V trace, but there was still a path with weak resistance between them. Measurements on the input pin of the L7806CD2T-TR were performed to determine if the short was between the regulators input and output.

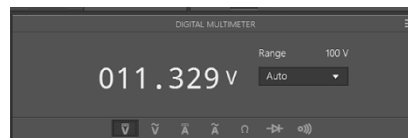


Figure 37 Measurement of the 6V regulator input

The design of the 6V regulator is based off the illustration in Figure 38, which was provided by the regulator's datasheet. The cause of the short was identified to be the transistors. Q1 is supposed to allow current to flow to V0 to allow greater current than the regulator can supply, but if the voltage drop across the sense resistor, R_{SC}, gets high then Q2 is meant to open and turn off Q1. This would redirect the excess current causing the short to the internal short circuit protection of the regulator. Unfortunately, Q1 needed to have a higher Base-Emitter threshold voltage while Q2 needed to have a lower one.

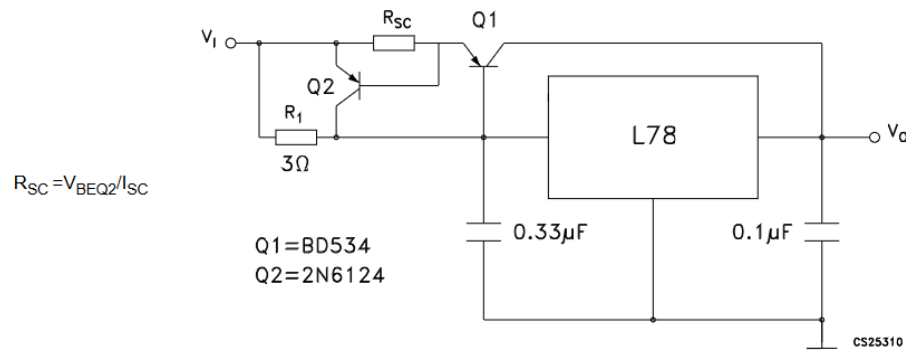


Figure 38 L78 Regulator Circuit Inspiration

Removing Q1 removed the short on the circuit. The output of the 6V regulator was measured again to check whether removing the transistor Q1 fixed the issue. This had fixed the

issue as the voltage measured in Figure 39 went down to the proper 6V that was expected as the output of the regulator.

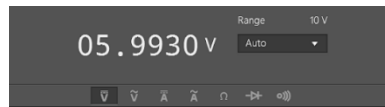


Figure 39 Double Checked 6V Regulator Voltage

After fixing the transistor issue, the voltages powering the MSP342 were tested. These test points did provide voltages of 3.3 Volts and 5 volts but were later realized to not be enough to power the double logic circuitry of the MSP432. This resulted in the switch to USB 2.0 charging rather than PCB charging.

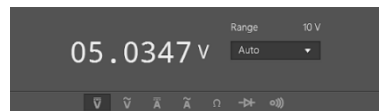


Figure 40 5V Jumper Voltage

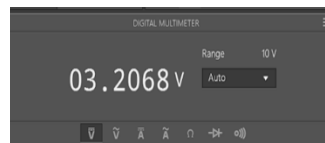


Figure 41 3.3V Jumper Voltage

After testing the parts individually, the parts were tested together and when the parts were tested together there was an issue with noise impacting the signal sent to the servo causing it to move randomly and uncontrollably. When this problem was found, the chassis was not built, and wires were not properly spaced so the thought was that better wire management was going to solve the problem. After the chassis was built the system was tested again and the noise persisted so the servo signal was measured with the motors running and without the motors running shown in Figure 42 and 44.



Figure 42 Servo Signal without Motor Running



Figure 43 Servo Signal with Motor Running

After looking at the noise a low pass filter was used in the attempt to not let the high-frequency noise impact the servo. However, the lowpass filter changed the square wave signal into the servo too much causing the servo to not be able to use the signal. The lowpass filter caused the output of the signal to become a sawtooth instead of the desired square wave signal. This can be seen in Figure 44.



Figure 44 Servo Signal with Lowpass Filter

Instead of using a lowpass filter, a voltage divider was used to reduce the level of the noise to an area that would not impact the servo while keeping the signal to the servo at an acceptable level for operation. The signal after the voltage divider can be seen in Figure 45.



Figure 45 Servo Signal with Voltage Divider

The PWM frequency for the motors was selected at 333Hz since it experimentally produced the least audible noise while operating at a duty cycle of 60%. The recommended operating frequency to minimize noise of the motor is 20kHz but operating at even a 10% duty cycle for such a high frequency drew more current than the wall adapter could safely provide. Additionally, testing the motors at higher frequencies appeared to increase the strain experienced by the servo, so it was deemed too risky to the higher frequencies for a minor reduction in audible noise.

Final Result

The result of the Decapacitors was a fully functioning automatic dog ball launcher that rotated on a 180-degree axis. The four main goals of functionality for this design were as

follows: the ability to rotate on a 180-degree axis, the ability to fire a ball at multiple speeds, the ability to Bluetooth connect and send signals through a webserver and creating a water-resistant chassis to help keep the electronics free of water as the device is intended for outdoor use. The servo mechanism that was designed supports the weight of the launching mechanism and can fully rotate on a 180-degree axis, allowing the motors to launch the ball at any angle desired. This completes the first category. Additionally, the device can rotate the motors at multiple speeds and launch the ball at various distances, which completes the second requirement. Third, the webserver can send Bluetooth signals to the microcontroller which receives those signals, translates the data and then sets the speed of the motors, the angle of the servo and the time in which the machine should run, which completes the third category of the rubric. The final category was that the launcher needed to have a water-resistant chassis. A chassis was constructed that attempts to be water-resistant, but the base of the structure and many of the supporting pieces were made of wood which was the simplest and most accessible material. Unfortunately, wood is not water-resistant, and would not classify for this section of the grading rubric. However, since a chassis was constructed, one point would still be awarded for this section. The overall grading table that defined this section is illustrated in Table 3 followed by the grade conversion in Table 3.

Table 3 Grading Rubric

	Tennis Ball Launching	Tennis Ball aiming	Web Application	Housing Design
2	Can launch ball with speed control	Can aim ball within 180 degrees	Can set angles and schedules	A Chassis that is water resistant
1	Can launch ball without speed control	Can aim ball within 90 degrees	Can set angles	A Chassis that is not water resistant
0	Does not launch ball	Cannot aim ball	Cannot communicate with app	No Chassis

Table 4 Grading Conversion

Points	Grade
8-7	A
6-5	B
4-3	C
2-0	D

Based on the previous discussion, the team believes that a 7 out of 8 should be awarded with the point missing being the non-water-resistant chassis. This would lead to an A on the letter grade scale.

While this is a good grade based on the original scale, there were still some additional goals that were not met by the team. The first one is that the timing mechanism was originally intended to schedule the machine to run later in the day. This feature would fulfil the original goal of allowing the owner to schedule a runtime during the day before they leave for work. This would allow their pet to have some form of entertainment that would alleviate some of the issues that they were experiencing throughout their time alone. Another goal that the group had that was not fulfilled was the launching mechanism, being able to fire tennis balls rather than racquetballs. The racquetball system was used due to the availability of CAD files that were found. This made it so that the system had to change to work with racquetballs. To accomplish this goal, the group would scale everything up in size to match a tennis ball. A final goal that was not met was uploading the Bluetooth API that Hayden developed into an Amazon Web Services server. Right now, AWS file dependencies are not updated to extend the need to upload the Bluetooth Python package.

Overall, the automatic ball launcher completed most of its core functionality that was required, but there were many additional goals that the group had that they did not meet. These additional goals would have optimized the system and made functionality better for the user, but there was not enough time to complete these items and they could be finished in the future.

Costs

The cost of one unit is \$245.29 however this does not include the cost of the chassis since we were allowed to print most of our chassis for free. The total cost breakdown for 1 unit is in Appendix D. When considering the cost of building 10000 units some parts did give discounts when ordered in bulk, however some did not have that information listed and it was assumed that the price would stay the same even in bulk buying, because of this the total price to build 10000 units is \$2,066,652.50. The part driving the price for 10000 units are the motor driver boards which are around 50 dollars a unit without any price decrease buying in bulk. So, if 10000 units were to be built the Driver boards would have to be replaced with a cheaper alternative.

Future Work

Improvements to the Scheduling System/Timing Mechanism

The original intention of the project when creating the idea for the timing mechanism was to allow the owner to send a signal for the device to store, and the device would then turn on later during the day. The owner would send a signal first thing in the morning at 8AM when they leave telling the device to turn on at 3PM for twenty minutes. The initial reasoning behind this thought process was that the launcher was made with separation anxiety and loneliness of dogs in mind based on the pandemic and all the owners beginning to return to long days of work. According to the American Animal Health Association, about 10% of all Americans adopted a dog in 2020. This would lead to an excessive number of dogs that have never experienced time alone from their owners due to the pandemic [64]. Separation anxiety is a severe problem that needs to be dealt with in pet health throughout the world. Improving the timing mechanism to be able to receive a signal that runs the system much later in the day would be a significant improvement based on the original intentions. This would allow the dog to have some form of entertainment when the owner is at work all day and alleviate some of the loneliness that they may be experiencing.

Improvements to the Chassis/Structural Integrity of the System

The system could be improved with more dedicated parts that were created using CAD rather than the wood structure that makes up most of the base now. This wood was used due to time constraints, and a more dedicated chassis that was better structurally planned out would make the system much sturdier and more secure. This would reduce vibrations, reduce the strain on the servo that supports the entire system for the rotation, and alleviate some of the noise in the signals that was being experienced. Additionally, implementing stronger connections and increasing the structural integrity would extend the life of the device and ensure that the servo was supporting the least weight possible, and everything was evenly distributed.

Additionally, the original vision for the automatic ball launcher was to launch a tennis ball. This means that a future improvement could be made in scaling the size of the system up to create room for a tennis ball launch rather than the racquetball that this system currently functions with. One final improvement to the structural design could come in the form of longer/more thorough 3D prints. The designs revolved around efficiency and had to match the capabilities of the 3D printers that were available in Clem and the Mech lab. If the printers were much more powerful, the designs could have been much thicker and more structurally sound.

Web Application Change to Wi-Fi Connection

An improvement for the web application is hosting the application on a domain and using a Wi-Fi connection with the microcontroller. The current working version is hosted on the local host and connects to the microcontroller via a Bluetooth API. A Bluetooth connection limits the application user to be near the microcontroller and the system itself. With a Wi-Fi connection between the web application and the microcontroller, the launcher can be run and set at anytime from anywhere with Wi-Fi access. Any device with access to the internet will be able to connect to the microcontroller. Another improvement is to have an actual mobile application with a Wi-Fi connection instead of a web service that can be accessible via a mobile device. A mobile application would serve to be more user friendly, and the mobile application would ensure the internet connection exists whether it is via Wi-Fi or cellular data.

Randomization of Mechanics with a Sensor

One way to make the system more interactive for dogs would be to implement a sensor in the launch tube that would alert the microcontroller whenever a ball has passed through for numerous reasons. The first item that could be addressed with this sensor implemented would be randomization of the system. The speed of the motors and the position of the servo could be varied with a simple method every time the sensor is pressed by the ball during run time, which would add more interactivity to the system for the dog to play with. Moreover, the sensor would allow the user to see data about how many times the ball was launched during the uptime of the machine. This data may illustrate if there were certain settings that made the dog utilize the system the most, if there were certain times of day when the dog was the most active with the machine or any other items that may be important to note.

References

- [1] K. Kavin, "Dog adoptions and sales soar during the pandemic," *The Washington Post*, 15-Aug-2020. [Online]. Available: <https://www.washingtonpost.com/nation/2020/08/12/adoptions-dogs-coronavirus/>. [Accessed: 17-Dec-2021].
- [2] "Evolution: Artificial selection and domestication," *OpenLearn*. [Online]. Available: <https://www.open.edu/openlearn/nature-environment/natural-history/evolution-artificial-selection-and-domestication/content-section-3.1>. [Accessed: 14-Sep-2021].
- [3] "U.S. pet ownership statistics," *American Veterinary Medical Association*. [Online]. Available: <https://www.avma.org/resources-tools/reports-statistics/us-pet-ownership-statistics>. [Accessed: 17-Nov-2021].
- [4] M. Kearl, "Alone time for dogs: How much is too much?," *American Kennel Club*, 29-Apr-2021. [Online]. Available: <https://www.akc.org/expert-advice/training/alone-time-dogs-much-much/>. [Accessed: 14-Sep-2021].
- [5] "Separation anxiety," *ASPCA*. [Online]. Available: <https://www.asPCA.org/pet-care/dog-care/common-dog-behavior-issues/separation-anxiety>. [Accessed: 14-Sep-2021].
- [6] "Automatic dog ball launcher - ifetch and idig dog toys," *iFetch Dog Toys - Automatic Dog Ball Launcher*, 26-May-2021. [Online]. Available: <https://goifetch.com/>. [Accessed: 14-Sep-2021].
- [7] "iDogmate small automatic pet dog ball thrower - tennis ball launcher - puppy or small dog fetch - ball throwing machine - pet dog toy with 1.75," *iDogmate*. [Online]. Available: <https://idogmate.com/collections/frontpage/products/idogmate-automatic-dog-ball-thrower-pet-toy>. [Accessed: 14-Sep-2021].
- [8] "MSP430G2553 | MSP430G2x/i2x | MSP430 ultra-low-power MCUs | Description & parametrics." [Online]. Available: <http://www.ti.com/product/MSP430G2553>. [Accessed: 06-Dec-2016].
- [9] "Boostxl-CC2650MA," *BOOSTXL-CC2650MA Evaluation board | TI.com*. [Online]. Available: <https://www.ti.com/tool/BOOSTXL-CC2650MA>. [Accessed: 15-Dec-2021].
- [10] "PLA plastic / material – the ultimate guide," *All3DP*, 28-Oct-2021. [Online]. Available: <https://all3dp.com/2/what-is-pla-plastic-material-properties/>. [Accessed: 17-Dec-2021].
- [11] "Why is plastic harmful? – plastic pollution coalition." [Online]. Available: <https://plasticpollutioncoalition.zendesk.com/hc/en-us/articles/222813127-Why-is-plastic-harmful->. [Accessed: 17-Sep-2021].
- [12] Admin, "Deforestation - causes, effects, control of deforestation with videos and faqs," *BYJUS*, 07-Oct-2021. [Online]. Available: <https://byjus.com/chemistry/deforestation/>. [Accessed: 15-Dec-2021].
- [13] APPA, "Law library article," *American Pet Products Association*, 2021. [Online]. Available: https://www.americanpetproducts.org/law/lawlibrary_article.asp?topic=62. [Accessed: 08-Sep-2021].
- [14] "Phthalates: Final guidance on inaccessible component parts, 16 CFR part 1199," *U.S. Consumer Product Safety Commission*. [Online]. Available: <https://www.cpsc.gov/Regulations-Laws--Standards/Rulemaking/Final-and-Proposed-Rules/Phthalates>. [Accessed: 10-Dec-2021].
- [15] Webmaster, "Embedded C coding standard," *Barr Group Software Experts*, 26-May-2016. [Online]. Available: <https://barrgroup.com/embedded-systems/books/embedded-c-coding-standard>. [Accessed: 16-Dec-2021].

- [16] Ip66 rating: Requirements of waterproof ip66 rated enclosures,” *TechTalk Blog*, 16-Jul-2019. [Online]. Available: <https://www.polycase.com/techtalk/waterproof-electronic-enclosures/ip66-rating-requirements-of-waterproof-ip66-rated-enclosures.html>. [Accessed: 17-Sep-2021].
- [17] “Motors and generators,” *NEMA*. [Online]. Available: <https://www.nema.org/standards/view/motors-and-generators>. [Accessed: 17-Sep-2021].
- [18] “NFPA 70 (nec) ARTICLE 430: Understanding Motors, motor circuits, & CONTROLLERS,” *Grace Technologies*. [Online]. Available: <https://www.graceport.com/help/nfpa-70-nec-article-430-understanding-motors-motor-circuits-controllers>. [Accessed: 17-Sep-2021].
- [19] Webmaster, “Embedded C coding standard,” *Barr Group Software Experts*, 26-May-2016. [Online]. Available: <https://barrgroup.com/embedded-systems/books/embedded-c-coding-standard>. [Accessed: 16-Dec-2021].
- [20] “IEEE 802.15.1-2002 - IEEE standard for telecommunications and information exchange between systems - LAN/man - specific requirements - part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (wpans),” *IEEE SA - The IEEE Standards Association - Home*. [Online]. Available: https://standards.ieee.org/standard/802_15_1-2002.html. [Accessed: 17-Dec-2021].
- [21] “MSP-EXP432P401R,” *MSP-EXP432P401R | Buy TI parts | TI.com*. [Online]. Available: <https://www.ti.com/store/ti/en/p/product/?p=MSP-EXP432P401R>. [Accessed: 15-Dec-2021].
- [22] “RSLK Max Edition curriculum,” *TI*. [Online]. Available: <https://university.ti.com/en/faculty/ti-robotics-system-learning-kit/ti-rslk-max-edition-curriculum>. [Accessed: 15-Dec-2021].
- [23] “General python FAQ,” *General Python FAQ - Python 2.7.18 documentation*. [Online]. Available: <https://docs.python.org/2.7/faq/general.html#what-is-python>. [Accessed: 15-Dec-2021].
- [24] “Bleak,” *bleak*. [Online]. Available: <https://bleak.readthedocs.io/en/latest/index.html>. [Accessed: 15-Dec-2021].
- [25] Vxgmichel, “Vxgmichel/aioconsole: Asynchronous console and interfaces for Asyncio,” *GitHub*. [Online]. Available: <https://github.com/vxgmichel/aioconsole>. [Accessed: 15-Dec-2021].
- [26] long2ice, “FASTAPI admin,” *FastAPI Admin*. [Online]. Available: <https://fastapi-admin.github.io/>. [Accessed: 17-Dec-2021].
- [27] “Introduction,” *Uvicorn*. [Online]. Available: <https://www.uvicorn.org/>. [Accessed: 17-Dec-2021].
- [28] Microsoft, “Documentation for visual studio code,” *RSS*, 03-Nov-2021. [Online]. Available: <https://code.visualstudio.com/docs>. [Accessed: 15-Dec-2021].
- [29] Cianfrocco-Lab, “What is github? · Cianfrocco-Lab/Old-school-processing wiki,” *GitHub*. [Online]. Available: <https://github.com/cianfrocco-lab/Old-school-processing/wiki/What-is-GitHub%3F>. [Accessed: 15-Dec-2021].
- [30] “React – a JavaScript library for building user interfaces,” – *A JavaScript library for building user interfaces*. [Online]. Available: <https://reactjs.org/>. [Accessed: 15-Dec-2021].
- [31] “Bootstrap,” *React Bootstrap*. [Online]. Available: <https://react-bootstrap.github.io/>. [Accessed: 15-Dec-2021].

- [32] “Getting started,” *Getting Started | Axios Docs*. [Online]. Available: <https://axios-http.com/docs/intro>. [Accessed: 15-Dec-2021].
- [33] iFetch, LLC, “Automatic Ball Launcher,” 17-May-2016.
- [34] W. Dana, “Pet Operated Ball Thrower,” 03-October-2011.
- [35] M. Levin, G. Newsome, and R. Davidson, “Ball Thrower,” 03-May-2011.
- [36] Hi Tech, “HS-805BB Servo-Stock Rotation,” HS-805BB. Datasheet. <https://www.servocity.com/hs-805bb-servo/>
- [37] “GP,” Amazon, 2011. [Online]. Available: <https://www.amazon.com/gp/product/B08KXY5M91?tag=youtube-redirect-20&geniuslink=true>. [Accessed: 17-Dec-2021].
- [38] “Pololu G2 high-power motor driver 18V17,” Pololu Robotics & Electronics. [Online]. Available: <https://www.pololu.com/product/2991>. [Accessed: 17-Dec-2021].
- [39] “Amazon.com: Shnitpwr 12V 5A 60W power supply Adapter AC DC ...” [Online]. Available: <https://www.amazon.com/SHNITPWR-Converter-100V-240V-Transformer-5-5x2-5mm/dp/B07NR6FPN9>. [Accessed: 17-Dec-2021].
- [40] “GP,” Amazon, 2011. [Online]. Available: <https://www.amazon.com/gp/product/B0711Q5B49?tag=youtube-redirect-20&geniuslink=true>. [Accessed: 17-Dec-2021].
- [41] “Amazon.com: 5.5 X 2.1 mm 5A DC power Jack Socket Threaded ...” [Online]. Available: <https://www.amazon.com/Socket-Threaded-Connector-Adapter-3-94Ich/dp/B07LF1193N>. [Accessed: 17-Dec-2021].
- [42] “L7806CD2T-TR,” *DigiKey*. [Online]. Available: <https://www.digikey.com/en/products/detail/https://www.digikey.com/en/products/detail/stmicroelectronics/L7806CD2T-TR/1884018>. [Accessed: 17-Dec-2021].
- [43] “Zxtp25020dfhta,” *DigiKey*. [Online]. Available: <https://www.digikey.com/en/products/detail/diodes-incorporated/ZXTP25020DFHTA/1557752>. [Accessed: 17-Dec-2021].
- [44] “L7805ABD2T-TR,” *DigiKey*. [Online]. Available: <https://www.digikey.com/en/products/detail/stmicroelectronics/L7805ABD2T-TR/585694>. [Accessed: 17-Dec-2021].
- [45] “C1206C334K3RAC7800,” *DigiKey*. [Online]. Available: <https://www.digikey.com/en/products/detail/kemet/C1206C334K3RAC7800/416048>. [Accessed: 17-Dec-2021].
- [46] “C1206C104J1RAC7800,” *DigiKey*. [Online]. Available: <https://www.digikey.com/en/products/detail/kemet/C1206C104J1RAC7800/2215064>. [Accessed: 17-Dec-2021].
- [47] B. ThingVerse, “The Ultimate DIY Automatic dog ball launcher. with video tutorial. by brankly,” *Thingiverse*, 13-Oct-2021. [Online]. Available: <https://www.thingiverse.com/thing:5019572/files>. [Accessed: 16-Dec-2021].
- [48] B. Ray, “Bluetooth vs. Bluetooth Low Energy: What's the difference? [2021 update]: Blog: Link labs,” *Blog | Link Labs*, 19-Aug-2021. [Online]. Available: <https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy>. [Accessed: 16-Dec-2021].
- [49] “How to send data between PC and Arduino using Bluetooth Le,” *Ladvien's Lab*, 11-Jul-2020. [Online]. Available: <https://ladvien.com/python-serial-terminal-with-arduino-and-bleak/>. [Accessed: 16-Dec-2021].

- [50] “Serverless on AWS,” *Serverless Computing*. [Online]. Available: <https://aws.amazon.com/serverless/>. [Accessed: 16-Dec-2021].
- [51] *Heroku*. [Online]. Available: <https://dashboard.heroku.com/>. [Accessed: 16-Dec-2021].
- [52] “Launchxl-CC26X2R1,” *LAUNCHXL-CC26X2R1 Evaluation board | TI.com*. [Online]. Available: <https://www.ti.com/tool/LAUNCHXL-CC26X2R1>. [Accessed: 16-Dec-2021].
- [53] “CCSTUDIO,” *CCSTUDIO IDE, configuration, compiler or debugger | TI.com*. [Online]. Available: <https://www.ti.com/tool/CCSTUDIO>. [Accessed: 15-Dec-2021].
- [54] “Pololu - 30:1 micro metal gearmotor HP 6V with Extended Motor Shaft,” Pololu Robotics & Electronics. [Online]. Available: <https://www.pololu.com/product/2212>. [Accessed: 17-Dec-2021].
- [55] “Pololu - power HD high-torque Servo 1501MG,” Pololu Robotics and Electronics. [Online]. Available: <https://www.pololu.com/product-info-merged/1057>. [Accessed: 17-Dec-2021].
- [56] “Pololu - magnetic encoder pair kit with top-entry connector for Micro Metal gearmotors, 12 CPR, 2.7-18V,” Pololu Robotics & Electronics. [Online]. Available: <https://www.pololu.com/product/4760>. [Accessed: 17-Dec-2021].
- [57] “105090004,” DigiKey. [Online]. Available: <https://www.digikey.com/en/products/detail/seed-technology-co-ltd/105090004/10667518>. [Accessed: 17-Dec-2021].
- [58] “MSP432P401R: MSP432P401R launchpad using external 5 volt supply connection (not USB port),” MSP432P401R: MSP432P401R launchpad using external 5 volt supply connection (not USB port) - MSP low-power microcontroller forum - MSP low-power microcontrollers - TI E2E support forums. [Online]. Available: <https://e2e.ti.com/support/microcontrollers/msp-low-power-microcontrollers-group/msp430/f/msp-low-power-microcontroller-forum/619374/msp432p401r-msp432p401r-launchpad-using-external-5-volt-supply-connection-not-usb-port>. [Accessed: 17-Dec-2021].
- [59] Boosterpack Checker - Ti Cloud tools. [Online]. Available: <https://dev.ti.com/bpchecker/#/>. [Accessed: 17-Dec-2021].
- [60] J. 27, “How to use LightBlue®: The go-to ble development tool,” *Punch Through*, 04-Mar-2021. [Online]. Available: <https://punchthrough.com/lightblue-features/>. [Accessed: 16-Dec-2021].
- [61] Postman, “Introduction,” Postman Learning Center. [Online]. Available: <https://learning.postman.com/docs/getting-started/introduction/>. [Accessed: 16-Dec-2021].
- [62] “VirtualBench,” National Instruments. <https://www.ni.com/en-us/shop/hardware/products/virtualbench-all-in-one-instrument.html> (accessed Dec. 08, 2020).
- [63] <https://img.hisupplier.com/var/userFiles/2008-09/28/155554.313.pdf>
- [64] “Separation anxiety and the ‘Pandemic puppy’: What lies ahead after lockdown,” American Animal Hospital Association. [Online]. Available: <https://www.aaha.org/publications/newstat/articles/2021-06/separation-anxiety-and-the-pandemic-puppy-what-lies-ahead-after-lockdown/>. [Accessed: 16-Dec-2021].

Appendix

Appendix A – Microcontroller Code

```
7 void motor_init(void)
8 {
9     // Right motor direction.
10    //
11    GPIO_setAsOutputPin(GPIO_PORT_P5, GPIO_PIN5);
12    // Left motor direction.
13    //
14    GPIO_setAsOutputPin(GPIO_PORT_P5, GPIO_PIN4);
15
16    // Sets the PWMS to low.
17    //
18    GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN6);
19    GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN7);
20
21    // Right motor sleep set to low.
22    //
23    GPIO_setAsOutputPin(GPIO_PORT_P3, GPIO_PIN6);
24    GPIO_setOutputLowOnPin(GPIO_PORT_P3, GPIO_PIN6);
25
26    // Left motor sleep set to low.
27    //
28    GPIO_setAsOutputPin(GPIO_PORT_P3, GPIO_PIN7);
29    GPIO_setOutputLowOnPin(GPIO_PORT_P3, GPIO_PIN7);
30 }
```

Figure 46 Motor Initialization Code

```
void motor_forward(uint16_t left_duty, uint16_t right_duty)
{
    // Call PWMDutyRight and Left.
    //
    pwm_duty_right(right_duty);
    pwm_duty_left(left_duty);

    // Direction setting.
    //
    GPIO_setOutputLowOnPin(GPIO_PORT_P5, GPIO_PIN5);
    GPIO_setOutputLowOnPin(GPIO_PORT_P5, GPIO_PIN4);

    // Motor enabling.
    //
    GPIO_setOutputHighOnPin(GPIO_PORT_P3, GPIO_PIN6);
    GPIO_setOutputHighOnPin(GPIO_PORT_P3, GPIO_PIN7);
}
```

Figure 47 Method to Move the Motors Forward


```

1 void pwm_init(uint16_t period, uint16_t duty_three, uint16_t duty_four)
2 {
3     // Sets the pins as outputs for the capture compares.
4     //
5     GPIO_setAsPeripheralModuleFunctionOutputPin(GPIO_PORT_P2,
6         GPIO_PIN6 | GPIO_PIN7, GPIO_PRIMARY_MODULE_FUNCTION);
7
8     // Sets the clock period.
9     //
10    up_down_config.timerPeriod = period;
11
12    // Sets the compare values for the duty cycles.
13    //
14    pwm_configure_three.compareValue = ((100 - duty_three) *
15        up_down_config.timerPeriod) / 100;
16    pwm_configure_four.compareValue = ((100 - duty_four) *
17        up_down_config.timerPeriod) / 100;
18
19    // Configures the timer.
20    //
21    Timer_A_configureUpDownMode(TIMER_A0_BASE, &up_down_config);
22
23    // Starts the counter.

```

Figure 48 Method to Initialize the Pulse Width Modulation Signals

```

1 /*
2  * @brief Updates the servo position.
3  *
4  * @param Servo Which servo to update.
5  */
6 @return Void.
7 */
8 void update_servo_position(ServoType *servo)
9 {
10     uint16_t next_compare_capture_value = 0;
11
12     next_compare_capture_value = (uint16_t) (SLOPE *servo->angle + INTERCEPT); // WARNING: cast to uint16_t.
13
14     if (next_compare_capture_value < INTERCEPT) {
15         next_compare_capture_value = (uint16_t) INTERCEPT; // WARNING: cast to uint16_t.
16     }
17     else if (next_compare_capture_value > UPPER_LIMIT) {
18         next_compare_capture_value = (uint16_t) UPPER_LIMIT; // WARNING: cast to uint16_t.
19     }
20
21     // Units of 2 microseconds.
22     //
23     *(servo->compare_capture_register_n) = next_compare_capture_value;
24 }

```

Figure 49 Method to Update the Servo Position

```

1 int main(void)
2 {
3     //This section declares all variables that are used throughout the process
4     // and initializes all the needed pins.
5     //
6     uint32_t h = 0;
7     uint8_t response_needed = 0;
8     uint16_t servo_position = 102;
9     uint16_t motor_speed = 0;
10    uint32_t time_variable = 0;
11    EnableInterrupts();
12    UART0_Init();
13    clock_init();
14    motor_init();
15    pwm_init(6000, 0, 0);
16    timer_atwo_init(&TaskSchedulerTimerA2ISR, TIMER_A2_PERIOD);
17    LaunchPad_Init();
18    initialize_servo_port_pins();
19    initialize_servo(&tilt, 0, (uint16_t*) (&(TIMER_A2->CCR[2]])); // Casted to uint16_t, careful of overflow
20    set_servo_position(&tilt, servo_position);
21    BLE_Init();

```

```

while (1)
{
    // This section checks to see if the AP has received a Bluetooth message, echos a response
    // and then checks the separate variables of the response.
    //
    if (AP_RecvStatus())
    {
        if (AP_RecvMessage(recv_buf, RECVSIZE)==APOK)
        {
            AP_EchoReceived(APOK);
            if ((recv_buf[3] == 0x55) && (recv_buf[4] == 0x88))
            {
                h = (recv_buf[8] << 8) + recv_buf[7];
                response_needed = recv_buf[9];
                // This section splits the receive buff into multiple values and then
                // prints the two that are relevant. It then sets the motor speed
                // and servo based on recv_buf[12] and the time on recv_buf[13].
                //
                if (h == handle_leds)
                {
                    if((recv_buf[12] & 0x0F) == 0x0a)
                    {
                        servo_position = 5;
                    }
                    if((recv_buf[12] & 0x0F) == 0x0b)
                    {
                        servo_position = 102;
                    }
                    if((recv_buf[12] & 0x0F) == 0x0c)
                    {
                        servo_position = 180;
                        //Setservo_position(&tilt, 180);
                    }
                    if(recv_buf[12] == 0x00)
                    {
                        servo_position = 102;
                        motor_speed = 0;
                    }
                    if ((recv_buf[12] & 0xF0) == 0xa0)
                    {
                        motor_speed = 40;
                    }
                }
                if ((recv_buf[12] & 0xF0) == 0xb0)
                {
                    motor_speed = 50;
                }
                if ((recv_buf[12] & 0xF0) == 0xc0)
                {
                    motor_speed = 60;
                }
                if (recv_buf[13] == 0x0a)
                {
                    time_variable = 5;
                }
                if (recv_buf[13] == 0x0b)
                {
                    time_variable = 10;
                }
                if (recv_buf[13] == 0x0c)
                {
                    time_variable = 15;
                }
            }
            // This section checks to see if a response is needed and sends a message if so.
            //

```

```

        if (response_needed)
        {
            AP_SendMessage(NPI_WriteConfirmationMsg);
            AP_EchoSendMessage(NPI_WriteConfirmationMsg);
        }
    }
}
if (time_variable != 0)
{
    OutValue("\n\rMotor Speed = ", motor_speed);
    OutValue("\n\rServo Position = ", servo_position);
    OutValue("\n\rTime Variable = ", time_variable);
    set_servo_position(&tilt, servo_position);
    Clock_Delay1ms(1000);
    motor_forward(motor_speed, motor_speed);
    timer_athree_init();
    // Timing section based on our three time options
    //
    while(revised_timer <= time_variable) {
        revised_timer = g_timer/40;
    }
    UART0_OutString("Done");
    motor_forward(0, 0);
    set_servo_position(&tilt, 102);
    timer_athree_stop();
    // Resetting variables.
    //
    revised_timer = 0;
    time_variable = 0;
    g_timer = 0;
    motor_speed = 0;
    servo_position = 0;
}
}
}

/** end of file */

```

Figure 50 Main Code

Appendix B – Bluetooth Software Code

```
class Item(BaseModel):
    distance: str
    time: str
    direction: str

@app.post("/items/")
async def create_item(item: Item):
    global in_dis
    global in_time
    global in_dir
    global loop
    global count
    loop = asyncio.get_event_loop()
    in_dis = item.distance
    in_time = item.time
    in_dir = item.direction
    try:
        connection = Connection(
            loop, read_characteristic, write_characteristic)
        asyncio.ensure_future(connection.manager())
        asyncio.ensure_future(user_console_manager(connection))
    except KeyboardInterrupt:
        print()
        print("User stopped program.")
    return in_dis + in_time + in_dir
```

Figure 51 Server URL Post Path

```

class Connection:

    client: BleakClient = None

    def __init__(
        self,
        loop: asyncio.AbstractEventLoop,
        read_characteristic: str,
        write_characteristic: str,
        #data_dump_handler: Callable[[str, Any], None],
        #data_dump_size: int = 256,
    ):
        self.loop = loop
        self.read_characteristic = read_characteristic
        self.write_characteristic = write_characteristic
        #self.data_dump_handler = data_dump_handler

        self.last_packet_time = datetime.now()
        #self.dump_size = data_dump_size
        self.connected = False
        self.connected_device = None

        self.rx_data = []
        self.rx_timestamps = []
        self.rx_delays = []

```

Figure 52 Bleak Connection Class

```

async def manager(self):
    print("Starting connection manager.")
    while True:
        if self.client:
            await self.connect()
        else:
            await self.select_device()
            await asyncio.sleep(15.0, loop=loop)

```

Figure 53 Connection Manager Function

```

async def connect(self):
    if self.connected:
        return
    try:
        await self.client.connect()
        self.connected = await self.client.is_connected()
        if self.connected:
            print(f"Connected to {self.connected_device.name}")
            self.client.set_disconnected_callback(self.on_disconnect)
            print(f"set_disconnected")
            #await self.client.start_notify(
            #    self.read_characteristic, self.notification_handler,
            #)
            #print(f"start_notify")
            while True:
                if not self.connected:
                    break
                await asyncio.sleep(3.0, loop=loop)
        else:
            print(f"Failed to connect to {self.connected_device.name}")
    except Exception as e:
        print(e)

async def select_device(self):
    print("Bluetooth LE hardware warming up...")
    await asyncio.sleep(2.0, loop=loop) # Wait for BLE to initialize.
    devices = await discover()

    # print("Please select device: ")
    diction = []
    num = 0
    for i, device in enumerate(devices):
        print(f"{i}: {device.name}")
        manudata = device.metadata["manufacturer_data"].keys()
        uuids = device.metadata["uuids"]
        print(uuids)
        print(manudata)
        if 13 in manudata :
            break

    self.connected_device = device
    self.client = BleakClient(device.address, loop=self.loop)
    Addr = device.address

```

Figure 54 Connection Connect & Select Device Function

```

async def user_console_manager(connection: Connection):
    i = 0
    while i == 0:
        if connection.client and connection.connected:
            input_str = in_dis + in_dir + "0" + in_time
            print(input_str)

            await connection.client.write_gatt_char(write_characteristic, bytes.fromhex(input_str))
            print(f"Sent: {input_str}")
            i = i + 1
        try:
            connection.client.disconnect()
        except :
            print("Disconnecting...")
    else:
        await asyncio.sleep(2.0, loop=loop)

```

Figure 55 User Console Manager

```
if __name__ == "__main__":  
    uvicorn.run(app, host="172.25.137.205", port=8000)
```

Figure 56 Server Run Function

Appendix C – Web Application Code

```
// hooks for each of the fields  
const [distance, setDistance]=useState('');  
const handleDistance=(event)=>{  
    console.log(event);  
    setDistance(event)  
}  
const [direction, setDirection]=useState('');  
const handleDirection=(event)=>{  
    console.log(event);  
    setDirection(event)  
}  
const [time, setTime]=useState('');  
const handleTime=(event)=>{  
    console.log(event);  
    setTime(event)  
}
```

Figure 57 Distance, Direction, and Time Hooks

```
const [submitted, setSubmitted] = useState(false);  
  
const [valid, setValid] = useState(false);  
  
const handleSubmit = (event) => {  
    event.preventDefault();  
    console.log(event.data);  
    if(direction && distance && time) {  
        setValid(true);  
    }  
    setSubmitted(true);  
};
```

Figure 58 Submit Button Hooks

```
function createPost() {
  axios
    .post("http://172.25.137.205:8000/items", {
      distance: distance,
      direction: direction,
      time: time
    })
    .then(response =>
      console.log(response)
    );
}
```

Figure 59 POST Request Function

Appendix D - Cost

1 Units				10000 Units			
amount	item	price per unit	Total	amount	price per unit	Total	
1	VoltageRegulator, L7806CD2T-TR	0.96	\$0.96	10000	\$0.53	\$5,298.80	
1	RESISTOR, 2.7Ω	0.59	\$0.59	10000	\$0.11	\$1,142.40	
1	RESISTOR, 3Ω	0.59	\$0.59	10000	\$0.11	\$1,142.40	
2	BJT_PNP, ZXTP25020DFHTA	0.6	\$1.20	20000	\$0.23	\$4,500.00	
2	CAPACITOR, 0.33μF	0.30	\$0.60	20000	\$0.05	\$994.80	
2	CAPACITOR, 0.1μF	0.76	\$1.52	20000	\$0.18	\$3,505.00	
1	VoltageRegulator, L7805ABD2T-TR	0.88	\$0.88	10000	\$0.42	\$4,247.20	
1	TERMINAL_BLOCKS, 282834-2	1.84	\$1.84	10000	\$0.74	\$7,351.70	
2	HEADERS_TEST, 1-534206-0	4.94	\$9.88	20000	\$2.86	\$57,246.00	
1	HEADERS_TEST, HDR1X6	0.52	\$0.52	10000	\$0.23	\$2,327.00	
1	TERMINAL_BLOCKS, 282834-5	3.38	\$3.38	10000	\$1.55	\$15,542.20	
1	printed circuit board	0.456	\$0.46	10000	\$0.46	\$4,560.00	
2	G2 High-Power Motor Driver18v17	49.95	\$99.90	20000	\$49.95	\$999,000.00	
2	775 DC Motor	30.99	\$30.99	20000	\$0.00	\$0.00	
1	HS-805BB Servo-Stock Rotation	46.99	\$46.99	10000	\$46.99	\$469,900.00	
1	MSP-EXP432P401R	19.99	\$19.99	10000	\$19.99	\$199,900.00	
1	BOOSTXL-CC2650MA	29	\$29.00	10000	\$29.00	\$290,000.00	
Total cost			\$249.29			\$2,066,652.50	

Figure 60 Cost for 1 unit (left) Cost for 10000 units (right)